# Optimal reassuranse av totalskader

av
Navdeep Singh Malhi

Masteroppgave i studieprogrammet
Modellering og Dataanalyse
med studieretning Finans, Forsikring og Risiko

**2015**


Veileder: Erik Bølviken

Det matematisk-naturvitenskapelige fakultet
Universitetet i Oslo
Mai 2015

# Sammendrag

Denne avhandlingen handler om optimal reforsikring. Reforsikring eller reassuranse er en type forsikring som forsikrer et forsikringsselskap. Et forsikringsselskap vil beskytte seg mot fremtidige store krav ved å overføre deler av risikoen over til andre forsikringsselskaper som kalles reassurandører. I dagens praksis brukes det ingen optimeringsmetoder for å finne ut av hvor mye risiko man skal overføre til reassurandør. Det er i noen selskaper vist at man sammenlikner en portefølje med og uten reassuranse samtidig som man ser på deres fortjeneste. Man lager deretter forskjellige alternativer som man velger ut ifra. Jeg vil forsøke å gi en utredning om reassuranse-kontrakter som optimerer visse kriterier som kan være interessante for selskaper å se på. Dette er kriterier som vil finne den beste balansen mellom gevinst og nedsiderisiko. Jeg vil deretter vise at disse kriteriene optimeres for de gitte reassuranse-kontraktene. Det blir deretter interessant å se på hvordan disse kontraktene endrer sine optimale former for gitte modeller og porteføljer. Sammenlikning av optimal reassuranse mot ingen reassuranse vil også gjøres. Det er blitt skrevet om optimal reassuranse i tidligere litteratur, men disse artikkelene er sterkt matematisk preget med lite fokus på modellering. Jeg vil anvende denne teorien, tilføye oppskrifter på algoritmer og simuleringer slik at optimal reassuranse kan settes ut i praksis. Utfordringene i forhold til modellering vil være i hovedfokus. Derfor blir det også viktig å se på konsekvensene av estimeringsavvikene knyttet opp mot disse beregningene.

# Forord

Denne masteravhandlingen er skrevet som en del av mastergraden i Modellering og dataanalyse med studieretning Finans, forsikring og risikoanalyse. Den utgjør 60 poeng av mastergraden. Temaet er optimal reassuranse av totalskader, hvilket jeg syns er et veldig spennende tema innenfor skadeforsikring.

Jeg vil takke min veileder Erik Bølviken for et godt samarbeid, lærerike samtaler og konstruktive tilbakemeldinger. Hans støttende og oppmuntrende veiledningsstil har vært veldig hjelpsom gjennom hele prosessen. Jeg vil takke Nils Haavardson for innspill og informasjon om dagens praksis innenfor reassuranse-beregninger.

Jeg vil også takke min familie for deres støtte og motivasjon. Min mor, Mohinder Kaur, og min søster, Amandeep Kaur, har vært der gjennom hele studietiden. Til slutt vil jeg takke min forlovede Ramanjit Kaur. Hun har vært en grunnpillar for meg i form av inspirasjon, motivasjon og kjærlighet. Hennes ubegrensede støtte har vært uvurderlig. Jeg takker ydmykt for all hjelp.

# Innhold

# Tabeller

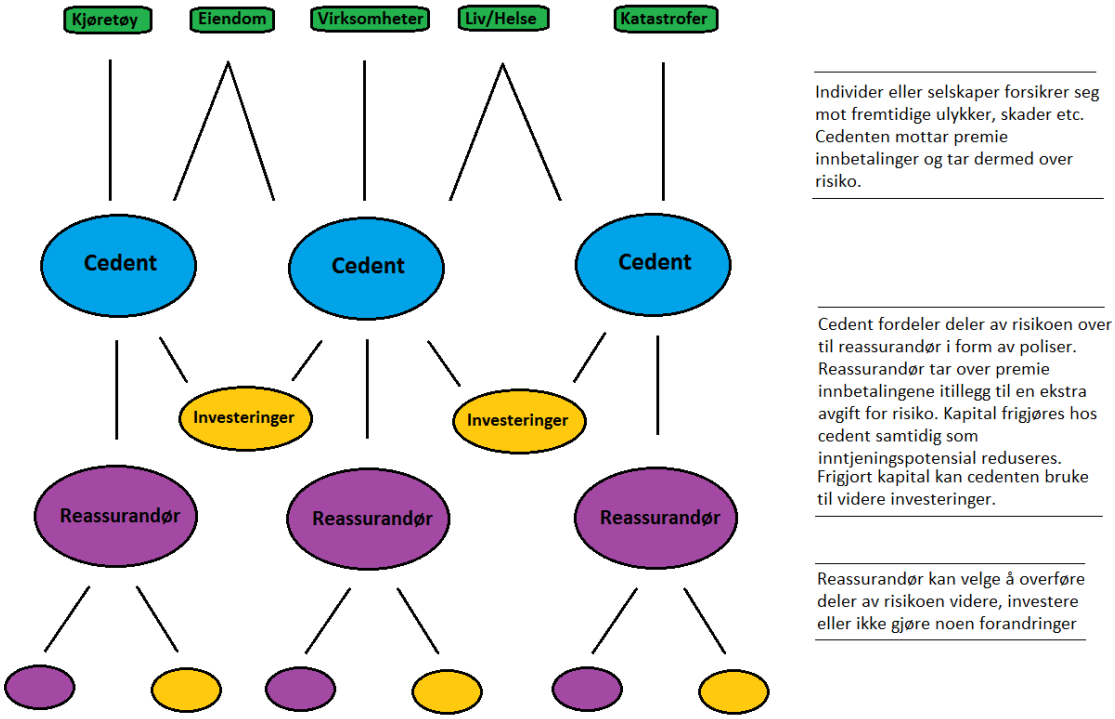


# Figurer

# 1 Innledning

## 1.1 Hva er reassuranse?

Reassuranse er et viktig og essensielt tema innenfor forsikring og økonomi. Reassuranse er uformelt sagt, en forsikring for forsikringsselskaper. Reassuranse hjelper forsikringsselskaper å redusere risikoen for store tap som selskapet ikke har kapasitet til å stå til ansvar for, samt frigjøre kapital for videre finansielle investeringer. I de fleste land har forsikreren/forsikringsselskapet en minimum kapitalreserve i bakhånd i tilfelle store krav som har liten sannsynlighet skulle forekomme. Et godt eksempel på det er fallet av Twin Towers i 2001. I dette tilfellet kunne umulig bare ett selskap ha stå til ansvar for alle krav knyttet til fallet av Twin Towers. Dette ville ha ført til en uheldig og uløselig situasjon for både forsikreren og den forsikrede. Dette hadde ikke gagnet samfunnsøkonomien. Selskapet Swiss Re sto til ansvar for den største risikoeksponeringen knyttet til de to bygningene. 22 % av et mulig krav skulle falle på Swiss Re. Resten av risikoen var overført som reassuranse til deres datter-/samarbeidsselskaper. Risikoen ble dermed spredd utover flere selskaper. Dette minsket risikoen for mulige store krav. Samtidig frigjorde dette kapital som var knyttet opp som kapitalreserve.




Figur 1: *Reassuranse oversikt*

Samtidig som reassuranse fører til stabile resultater og økonomisk vekst for et selskap stimulerer det også indirekte finansiell vekst i samfunnsøkonomien og en trygghet i dagens risikolandskap. Forsikringsselskaper utgjør en stor del av det økonomien i et land grunnet deres store summer som investeres i det finansielle markedet. Reassuransen tillater selskaper å frigjøre kapital for finansielle investeringer, samtidig som det skaper en trygghet rundt store samfunnskatastrofer.

## 1.2 Andel reassuranse

Når forsikreren sprer risikoen til reassurandør vil de også ha rett på premieinnbetalingene for polisene de overtar. Det vil i tillegg være en pris for risikoen reassurandør tar på seg. Dette er

beregninger man må ta i betraktning. Hvor mye risiko skal man overføre til reassurandør? Skal man reassurere alt over et visst risikonivå som utgjør forsikrerens maksimale grense for mulig utbetaling? Som et eksempel kan det nevnes at dagens praksis i DNB er å se på fordelingen til porteføljens lønnsomhet med og uten reassuranse hvor man ser på balansen mellom egenkapital og inntjeningspotensial. Med reassuranse øker kapitalen, men inntjeningspotensialet reduseres. Dette anskueliggjøres i lys av lønnsomheten til porteføljen. Noen form for optimalisering av kontrakter foretas ikke. Alternativer testes og det beste alternativet velges. Før man kommer til disse beregningene brukes Monte Carlo for simulering. Men aller først må selskapet se på sin visjon og sine mål hvor flere forhold blir aktuelle å ta i betraktning når reassuranse skal kjøpes.

- Selskapets strategi. Skal selskapet vokse, bli mer lønnsomt eller en kombinasjon?
- Selskapets produktmiks (andel bilforsikring, villaforsikring, reiseforsikring etc.)
- Selskapets eiendomsmiks (andel aksjer, andel eiendom, andel obligasjoner etc.)
- Selskapets beholdning av egenkapital
- Generelt rentenivå i samfunnet
- Eieres preferanser (en eier kan foretrekke et mer forutsigbart resultat som over tid er mindre enn et resultat som svinger mer fra år til år, men som over tid er større)
- Markedet for reassuranse (priser på reassuransekontrakter etc.)

## 1.3 Problemstilling

Mitt fokus i denne avhandlingen skal være å se på reassuranse-kontrakter som optimerer kriterier forsikringsselskaper finner interessante. Alle beregningene i denne avhandlingen skal gjøres på totalskader. Kriteriet vi skal se på vil optimere forventet gevinst mot nedsiderisiko. Vi skal se på to kriterier for nedside risiko, vanlig value-at-risk og betinget value-at-risk. Det skal vises at disse kriteriene oppnår et maksimum for respektive reassuranse-kontrakter. Begrenset stop-loss og enkel stop-loss. Det blir interessant å se hvordan grensene i reassuranse-kontraktene endrer seg for forskjellige typer porteføljer og hvordan deres optimale kriterieverdier er sammenliknet opp mot situasjonen hvor det ikke er noe reassuranse til stede. Tilhørende reassuranse-kontrakter skal jeg gjøre rede for samtidig som jeg gir en innføring i hvordan kriteriene skal optimeres. Bootstrap og Monte Carlo skal brukes i simuleringprosessen. Det skal tas i bruk en praktisk tilnærming til estimeringen og deretter en sammenlikning opp mot de sanne verdiene. Det blir derfor interessant å se konsekvensene av estimeringsfeilene som vil oppstå i en praktisk situasjon.

# 2 Grunnleggende skadeforsikring

I denne seksjonen skal jeg gi en introduksjon til grunnleggende matematiske formuleringer for totalskader samt vise hvilke optimale kontrakter for reassuranse vi skal bruke i resten av avhandlingen. En referanse for optimal reassuranse er Cheung, Sung, Yung og Yam(2011) og, Chi og Tan(2011). De redegjør for flere optimale kontrakter under forskjellige kriterier og risikomål. Siden disse to artiklene er sterkt matematisk og analytisk preget vil ikke alle tilfellene være relevante for analysen da jeg først og fremst skal se på de praktiske anvendelsene av disse kriteriene og kontraktene. De numeriske utfordringene skal være i hovedfokus.

## 2.1 Totalskade

La $X$ være summen av alle kravene i en portefølje over en viss tidsperiode, slik at

$$X = \sum_{i=1}^{N} Z_i \tag{2.1}$$

hvor N er antall krav og $Z_1, Z_2, \ldots$ hvor store de er. Anta at N og $Z_1, Z_2, \ldots$ er uavhengig og $Z_1, Z_2, \ldots$ identisk fordelte. Under en reforsikringssituasjon fordeles risikoen mellom reassurandør og cedent. Det gjøres ved hjelp av kontrakter som kan gjelde for hvert individuelt krav eller på en hel portefølje.

$X^{ce} = X - X^{re}$ er nettoansvaret til cedenten etter at reassuransen sin del har blitt tatt med i regnestykket. Summen av cedentens og reassuransens ansvar utgjør det totale kravet. Videre har vi

$$X^{ce} + X^{re} = X^{ce} + R(X) = X$$

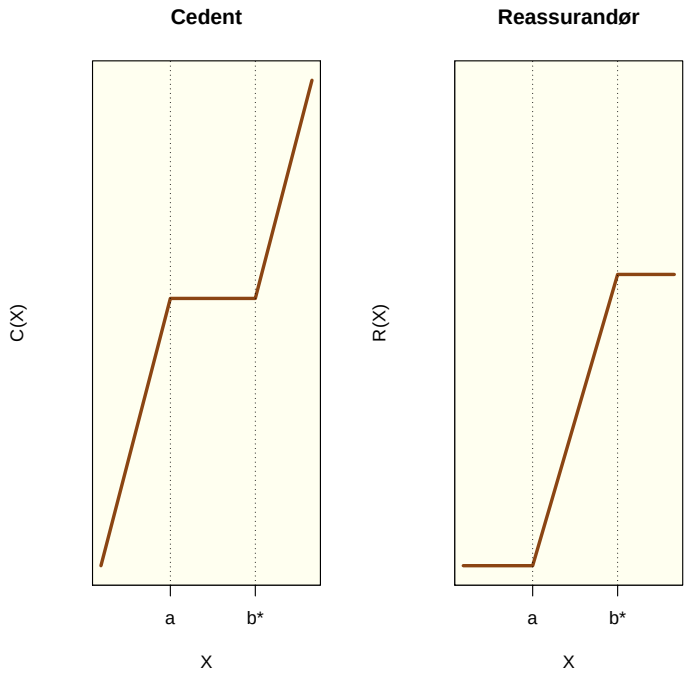
hvor R(X) er reforsikringskontrakten til reassurandør. Da kan man uttrykke cedentens ansvar slik

$$X^{re} = X - R(X) \tag{2.2}$$

R(X) er kontrakten som er av interesse. I delkapittel 3.1 vises det hvordan man kommer frem til optimale former av R(X). Den ene kontrakten vi kommer frem til er begrenset stop-loss som vises i figur 2. Den kan skrives på formen

$$R(X) = \min(\max((X - a)_+, b)$$

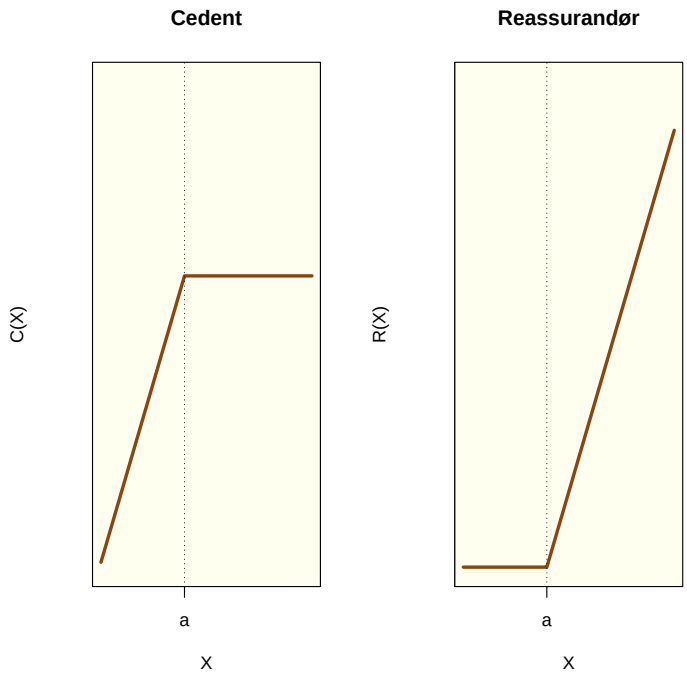
hvor $(X - a)_+ = \max(\text{X-a},0)$




Figur 2: *Begrenset stop-Loss*

I en begrenset stop-loss kan vi se at reassurandør ikke har noen bidrag hvis kravet X er mindre enn $a$. $a$ kan ses på som en nedre grense for reassurandør. De vil bidra med summen X-$a$ i intervallet mellom $a$ til $b^*$. For alle mulige X større enn $b^*$ vil reassurandør stå for en fast sum $b$, reassurandørs øvre grense. Cedenten vil stå for hele kravet X så fremt det er mindre eller lik a. Mellom intervallet $a$ og $b^*$ vil cedenten stå til ansvar for en fast sum $a$. Alle krav X over $b^*$ vil cedenten stå for summen X-$b$ da reassurandør kun betaler en fast sum $b$ for alle X>$b$.
Den andre kontrakten vi skal ta i bruk er enkel stop-loss. Denne kontrakten har kun èn parameter og skrives på formen

$$R(X) = (X - a)_+$$




Figur 3: *Enkel stop-loss*

Enkel stop-loss har ingen øvre grense for reassurandør. Cedenten står til ansvar for hele tapet på krav X når X$\leq$ $a$. Når kravet er større enn $a$ så er kravet til cedenten en fast sum $a$. Reassurandør tar på seg all risiko over $a$ med summen X-$a$. Enkel stop-loss reduserer risikoen til cedenten betraktelig siden de aldri står til ansvar for mer enn summen $a$. $a$ kan ses på som øvre grense for cedenten.

## 2.2 Porteføljer under gitte modeller

For å simulere data til videre bruk i avhandlingen vil jeg i det etterfølgende konstruere porteføljer under gitte sannynlighetsmodeller. Simuleringen består av 2 hoveddeler. Den første delen er antall krav og andre del er størrelsen på kravene. Hvor stor portefølje en bedrift vil ha eller hva en bedrift forsikrer vil påvirke størrelsen på disse to forskjellige delene. Fra praktisk forsikring vil dynamikken tilsi at det er forskjellige faktorer som påvirker disse delene. Antall skader har ingen påvirkning på størrelsen på skadene og omvendt. Derfor skal vi ta i bruk forskjellige uavhengige sannsynlighetsmodeller for hver del. Poisson-fordelingen og standard gamma-fordelingen vil bli brukt til å simulere antall krav. Størrelsen på skadene vil simuleres fra lognormal-, gamma- og pareto-fordelingene.

### 2.2.1 Poisson/lognormal-portefølje

Fra delkapittel 2.1 har vi at X er summen over alle skader $Z_i$ under en bestemt tidsperiode, hvor i = 1, 2,. . . , N. Vi antar at $Z_i$ og N er uavhengige. I dette delkapitelet skal vi se på tilfelle hvor N er poisson-fordelt med forventet antall krav $\lambda$ per tidsenhet T og der $Z_i$ er uavhengige lognormal-fordelte variabler. Her er $\lambda$=J$\mu$T, hvor J er antall poliser under risiko per tidsenhet og $\mu$ er kravintensitet per polise per tidsenhet.

Vi begynner med å definere poisson-fordelingen, se Devore og Berk(2007).
En stokastisk variabel X er poisson-fordelt med parameter $\lambda$, for z $\in \mathbb{N}$ hvis sannsynlighetsfordelingen er gitt ved

$$P(Z = z) = \frac{\lambda^z e^{-\lambda}}{z!}$$

E(Z) = Var(Z) = $\lambda$

Poisson-fordelingen er en diskret sannsynlighetsfordeling som modellerer antall forekomster av en gitt hendelse. Hendelser i vårt tilfelle er krav.

For simulering av kravstørrelse tar vi utgangspunktet fra en normal-fordelt stokastisk variabel, se Bølviken(2014).

$$Z^{`} = \alpha + \beta\epsilon, \quad \epsilon \sim N(0,1) \tag{2.3}$$

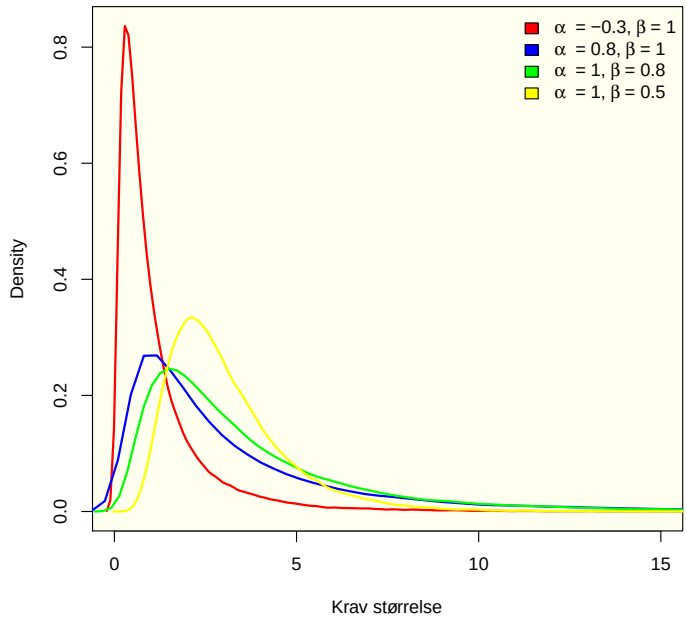
Vi vil jobbe på en logaritmisk skala og får lognormal-fordelingen

$$Z = e^{\alpha+\beta\epsilon,} \quad \epsilon \sim N(0,1) \tag{2.4}$$

med

$$E(Z) = e^{\alpha+\beta^2/2,} \quad og \quad sd(Z) = E(Z)\sqrt{e^{\beta^2} - 1} \tag{2.5}$$

hvor $Z^{`}$ = log(Z). Vi har nå en positiv vridd fordeling som egner seg godt til simulering av kravstørrelse. Her er $\alpha$ forventningen og $\beta$ standardavviket for fordelingen. For å få en konsekevent og oversiktlig notasjon i oppgaven vil jeg bruke $\alpha$ og $\beta$ som parametre i fordelingen for kravstørrelsene uansett sannsynlighetsmodell.




Figur 4: *lognormal-fordeling med gitte parameterverdier*

Forskjellige $\alpha$- og $\beta$-verdier vil gi oss forskjellige sannsynlighetsfordelinger. Det kan vi se eksempler på i figur 4. Dette vil gi grunnlag for forskjellige typer porteføljer. Den røde kurven i figur 4 representerer tilfellet hvor majoriteten av kravene er av små størrelser. Det er liten sannynlighet for store krav, noe som gjenspeiles i en smal hale. I motsetning til den grønne kurven som representerer parameterverdiene, $\alpha$=1 og $\beta$=0.8. Den fordelingen har en tyngre hale og dermed større sannsynlighet for at det kan oppstå store krav.

**Simulering**

Vi utvider notasjonen fra delkapittel 2.1 for å konstruere en poisson-/lognormal-portefølje. La oss anta at vi vil ha $m$ observasjoner av aggregerte krav $X$. Det vil si $X^* = X_1, X_2, \ldots, X_m$. Observasjon $j$ er definert ved $X_j$ som er relatert til sin $N_j$

$$X_j = \sum_{i=1}^{N_j} Z_i \tag{2.6}$$

hvor $N_j$ er uavhengig av $X_j$ og poisson-fordelt med parameter $\lambda$

$$N_j \sim poisson(\lambda) \tag{2.7}$$

videre er $\lambda$=J$\mu$T, mens $Z_i$ er uavhengig og lognormal-fordelt med parametre, $\alpha$ og $\beta$

$$Z_1, Z_2, \ldots, Z_{N_j} \sim lognormal(\alpha, \beta) \tag{2.8}$$

Simuleringen er en enkel algoritme hvor vi har 4 inputvariabler. Antall observasjoner $m$. $\lambda$ er forventet antall skader over en tidsperiode. $\alpha$ og $\beta$ er parametrene i normal-fordelingen.

**Algoritme 2.1** Poisson-/lognormal-portefølje

```
1   Inn: λ=Jμ T, m, α, β
2
3   for i = 1, 2, ..., m                    # m simuleringer
4       N = poisson(λ)                       # trekker antall skader
5       Z* = log-normal(N,α,β))              # Størrelse på hvert krav
5       X = sum(Z*)
6   return X*
```

For hver simulering trekker jeg $N$ antall krav fra poisson-fordelingen med parameter $\lambda$. Deretter simulerer jeg kravstørrelsen på N krav for hver replikasjon. Vi gjentar dette $m$ ganger og returnerer en vektor med med aggregerte krav, $X^* = X_1, X_2, \ldots, X_m$.

### 2.2.2 Poisson/gamma-portefølje

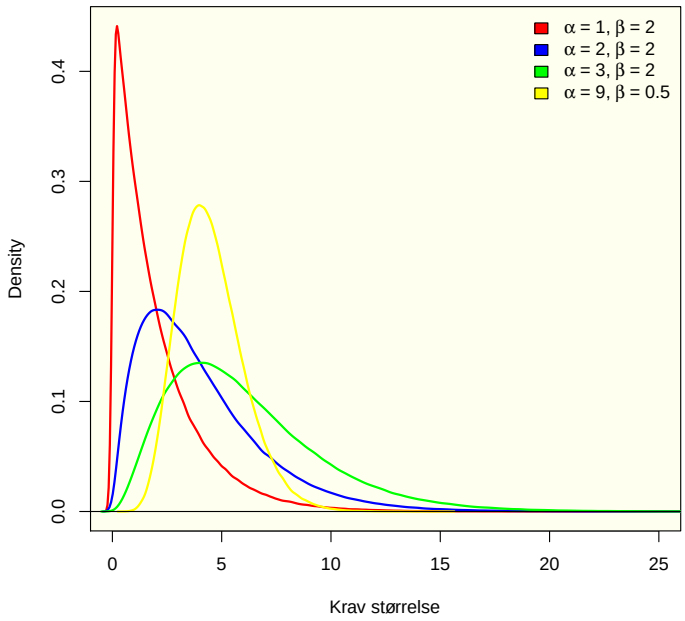
I denne seksjonen skal vi se på en poisson-/gamma-portefølje. Som definert i forrige seksjon vil $N$ fortsatt være poisson-fordelt. For simulering av kravstørrelse vil vi i denne seksjonen ta i bruk gamma-fordelingen, se Bølviken(2014).

En stokastisk variabel X er gamma-fordelt med parametrene $\alpha > 0$ og $\beta > 0$, for x $\in (0, \infty)$, hvis sannsynlighetsfordelingen er

$$f(x) = \frac{1}{\Gamma(\alpha)\beta^{\alpha}} x^{\alpha-1} e^{-\frac{x}{\beta}}$$

E(X)=$\alpha\beta$, Var(X) = $\alpha\beta^2$

Gamma-fordelingen er en positiv vridd fordeling som egner seg godt til simulering av kravstørrelse. Forskjellige verdier for $\alpha$ og $\beta$, hendholdsvis shape og scale, gir oss relativt forskjellige tetthetsfunksjoner som illustrert i figur 5. Gamma-fordelingens fleksibilietet i forhold til form, gjør den veldig nyttig for forskjellige scenarier innenfor eiendomsforsikring.




Figur 5: *Gamma-fordeling med gitte parameterverdier*

Vi har som nevnt 2 parametre i gamma-fordelingen. Disse påvirker formen på sannsynlighetsfordelingen til portefølje X, hvor hver skade $Z_i$ er gamma-fordelt. Vi ser fra figur 5 at vi kan ha en sannsynlighetsfordeling med forskjellige former. Tung hale, smal hale eller stor tyngde i starten er noen varianter vi kan nevne. Tung hale vil i praksis bety at vi har en portefølje med stor sannsynlighet for store krav og omvendt når man har en smal hale. Med parameter $\alpha$=1 og $\beta$=2 kan vi se stor tyngde i starten deretter et drastisk fall i sannsynligheten. Det representerer tilfellet hvor vi har en portefølje som med stor sannsynlighet vil bestå av mange små krav eller krav som er nær minimumet i størrelse.

**Simulering**

Vi utvider notasjonen fra delkapittel 2.1 for å konstruere en poisson-/gamma-portefølje. La oss anta at vi vil ha $m$ observasjoner av aggregerte krav $X$. Det vil si $X_1, X_2, \ldots, X_m$. Observasjon $j$ er definert ved $X_j$ som er relatert til sin $N_j$

$$X_j = \sum_{i=1}^{N_j} Z_i \tag{2.9}$$

hvor $N_j$ er uavhengig av $X_j$ og poisson-fordelt med parameter $\lambda$

$$N_j \sim poisson(\lambda) \tag{2.10}$$

videre er $\lambda$=J$\mu$T, mens $Z_i$ er uavhengig og gamma-fordelt med parametrene, $\alpha$ og $\beta$

$$Z_1, Z_2, \ldots, Z_{N_j} \sim gamma(\alpha, \beta) \tag{2.11}$$

Simuleringen er en enkel algoritme hvor vi har 4 inputvariable. Antall observasjoner $m$. $\lambda$ er forventet antall skader over en tidsperiode. $\alpha$ og $\beta$ er parametrene i gamma-fordelingen.

**Algoritme 2.2** Poisson-/gamma-portefølje

```
1   Inn: λ=Jμ T, m, α, β
2
3   for i = 1, 2, ..., m                    # m simuleringer
4       N = poisson(λ)                       # trekker antall skader
5       Z* = gamma(N,α,β))                   # Størrelse på hvert krav
5       X = sum(Z*)
6   return X*
```

For hver simulering trekker jeg $N$ antall krav fra poisson-fordelingen med parameter $\lambda$. Deretter simulerer jeg kravstørrelsen på N krav for hver replikasjon. Vi gjentar dette $m$ ganger og returnerer en vektor med med aggregerte krav, $X^* = X_1, X_2, \ldots, X_m$.

### 2.2.3 Poisson/pareto-portefølje

I denne seksjonen skal vi se på en poisson-/pareto-portefølje. Som definert i forrige seksjon vil $N$ fortsatt være poisson-fordelt, mens $Z_i$-ene i denne seksjonen skal være uavhengig pareto-fordelt.

De positive tilfeldige tallene $Z_i$ skal i denne seksjonen følge pareto-fordelingen, se Bølviken(2014). En stokastisk variabel X er pareto-fordelt når

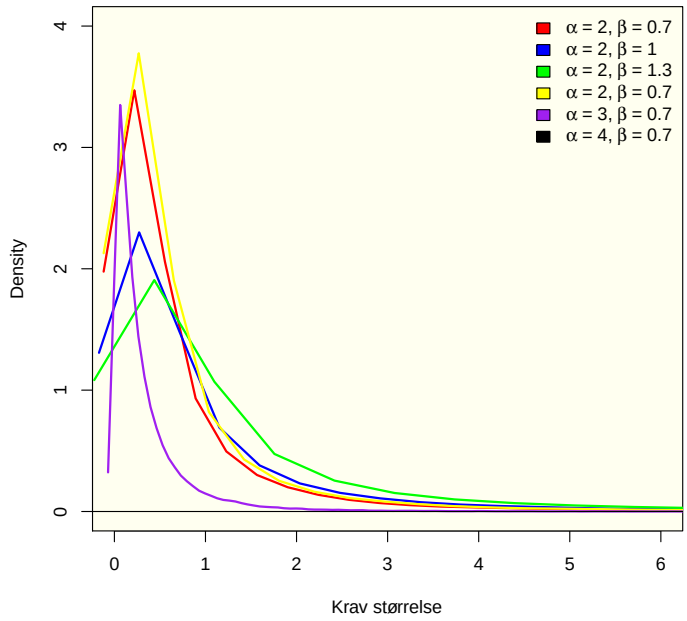
$$p(Z>z) = \frac{\alpha/\beta}{(1+z/\beta)^{1+\alpha}}$$

Grunnet noen av egenskapene til denne fordelingen må vi sette begrensninger på $\alpha$. Egenskapene er gitt ved

$$E(Z) = \frac{\beta}{\alpha - 1}$$

$$sd(Z) = E(Z)\sqrt{\frac{\alpha}{\alpha - 2}}$$

Siden $\alpha$-verdier under 2 gir oss uendelig varians og forventning vil vi bare se på $\alpha$-verdier over 2.




Figur 6: *Pareto-fordeling med gitte parameterverdier*

Vi har i figur 6 vist forskjellige sannsynlighetsfordelinger for pareto med forskjellige parameterverdier. Pareto-fordelingen er en tunghalet fordeling som egner seg godt til simulering av store krav i eiendomsforsikring. $\beta$ representerer minimumet for hvert krav og $\alpha$ er reduseringsraten. Jo større $\alpha$ jo fortere blir halen smalere.

**Simulering**
Det finnes ikke en innebygd funksjon i R for å generere tilfeldige tall fra pareto-fordelingen. Jeg må derfor ta i bruk inversmetoden. Jeg bruker sannsynlighetsfordelingen til pareto, F(Z) med invers Z = $F^{-1}(U)$. U er en uniform-fordelt variabel. Dette gir oss mulighetet til å generere tall fra pareto-fordeling ved hjelp av den uniforme fordelingen.

$$Z^* = \beta(U^{-1/\alpha} - 1) \tag{2.12}$$

hvor $U^*$ er en vektor med tilfeldig genererte tall fra den uniforme fordelingen. Vi skalerer variablen med parameteren $\beta$ og opphøyer med $-\alpha^{-1}$ som gir oss en vektor med tall $Z^*$. Denne vektoren består dermed av tilfeldig pareto-genererte tall.

Vi tar utgangspunkt fra samme notasjon som i gamma-porteføljen hvor vi bruker likning 2.3 og 2.4 som før, men bytter ut likning 2.5 med 2.7 som gjør kravstørrelsene pareto-fordelt.

$$Z_1, Z_2, \ldots, Z_N \sim pareto(\alpha, \beta) \tag{2.13}$$

Algoritmen i dette tilfelle har samme struktur som i en poisson-/gamma-portefølje.

**Algoritme 2.3** Poisson-/pareto-porteføljе

```
1   Inn: λ=Jμ T, m, α, β
2
3   for i = 1, 2, ..., m                    # m simuleringer
4       N = poisson(λ)                       # trekker antall skader
5       U = runif(N)[1]
6       Z* = β(U^{-1/α} − 1)                 # simulerer krav størrelse
7       X= sum(Z*)
7   return X*
```

### 2.2.4 Stokastisk portefølje

I denne seksjonen skal vi se på porteføljer med stokastisk intensitet. Jeg har til nå sett på situasjonen der poliser under risiko over en tidsperiode T har hatt en fast intensitet $\mu$. Dette gir oss en konstant intensitet. Det fins eksempler på kravintensiteter som varierer fra periode til periode innenfor skadeforsikring. Vi kan se på bilforsikring som et eksempel. Antall krav om sommeren og vinteren vil ikke ha samme forventning. Om vinteren vil glatt føre og dårlige veiforhold føre til at vi får en høyere kravintensitet, $\mu$. I sommertider vil veiforholdene være bedre og normalt sett vil vi se et fall i kravintensiteten. For å fange opp disse variasjonene fra periode til periode kan vi innføre en stokastisk intensitet.

**Stokastisk intensitet**
Settingen vil stort sett være den samme bortsett fra den konstante intensiteten $\mu$ i $\lambda$=J$\mu$T. Vi innfører en stokastisk variabel $\mu_{stok}$ i stedet.

$$N \sim Poisson(J\mu_{stok}T) \tag{2.14}$$

For å omgjøre $\mu$ til en stokastisk variabel vil jeg ta i bruk gamma-fordelingen, se Bølviken(2014).

$$\mu_{stok} = \xi G \qquad \text{hvor} \qquad G \sim Gamma(\psi) \tag{2.15}$$

Her er G en standard gamma-fordelt variabel med forventning 1 og standardavvik $\frac{1}{\alpha}$. Kombinerer vi dette med egenskapene til $\mu_{stok}$ får vi

$$E(\mu_{stok}) = \xi \qquad \text{og} \qquad sd(\mu_{stok}) = \xi/\sqrt{\psi} \tag{2.16}$$

hvor vi ser at $\mu_{stok}$ vil variere rundt $\xi$. Dette gir oss en stokastisk intensitet som varierer rundt en forventning. Størrelsen på fluktuasjonene avhenger av parameteren $\psi$

[1] runif er en innebygd funksjon i R som tar inn parameter N. Den genererer N antall tilfeldige tall fra den uniforme fordelingen.

**Algoritme 2.4** Poisson-/gamma-portefølje med stokastisk intensitet

```
1   Inn: m, α, β, ψ, ξ, J, T
2
3   for i = 1, 2, ..., m
4       μ_stok = ξG(ψ)                          # stokastisk intensitet
4       N = poisson(JμT)
5       Z* = gamma(N,α,β))
5       X = sum(Z*)
6   return X*
```

Algoritmen vil ha den samme strukturen som de tidligere porteføljene, men vi må tilføye en linje for å få den stokastiske effekten.

# 3 Optimal reassuranse

I denne seksjonen skal jeg redegjøre for optimal reassuranse under forskjellige sannsynlighetsmodeller. Første delkapittel skal omhandle optimal reassuranse på generell basis. Her skal det vises hvordan man kommer frem til de optimale formene for reassuranse-kontraktene som ble introdusert i delkapittel 2.1. Deretter er jeg interessert i å belyse hvordan optimalt reassuranse varierer med den underliggende modellen og deres parametre.

## 3.1 Optimalitet

En naturlig formulering av hva et forsikringsselskap er ute etter å studere er forventet fortjeneste i forhold til nedside-risiko. Dersom $\pi = \mathrm{E(X)}$ og $\pi^{re} = \mathrm{E}(X^{re})$ er forventet utbetaling brutto og forventet utbetaling av reforsikringen blir cedentens gevinst en gitt periode

$$G^{ce} = (1+\gamma)\pi + X^{re} - X - (1+\gamma^{re})\pi^{re} \tag{3.1}$$

der $\gamma$ og $\gamma^{re}$ er henholdsvis ladningene over for cedentens kunder og ved reforsikring. I praksis er $\gamma^{re} > \gamma > 0$. Det følger nå at

$$E(G^{ce}) = \gamma\pi - \gamma^{re}\pi^{re} \tag{3.2}$$

og et selskap vil prøve å gjøre denne så stor som mulig. Samtidig må selskapet ta hensyn til nedside-risiko. Et ofte anvendt kriterium er løsningen av likningen

$$Pr(X^{ce} > q_\epsilon^{ce}) = \epsilon \tag{3.3}$$

der $\epsilon$ er et gitt solvensnivå. Her er $q_\epsilon^{ce}$ den kapital cedent må avsette til dekning av fremtidige tap. Dette risikomålet uttrykker den finansielle risikoen til den bedrift. Bedrifter vil forsikre seg om at de begrenser risikoen til et nivå de vet de klarer å absorbere kravene som utgjør det verst tenkelige utfallet. Det verst tenkelige utfallet defineres på forhånd av bedriften som (1-$\alpha$)% reserven. Et alternativt kriterium, ofte fremhevet i den akademiske litteraturen er

$$C_\epsilon = E(X^{ce} | X^{ce} > q_\epsilon^{ce}) \tag{3.4}$$

kjent som betinget value-at-risk. Dette er gjennomsnittstapet som overskrider (1-$\alpha$)%-persentilen for kravet X. Dette kriteriet er mer sensitivt for formene på fordelingen ute i halene. Begge disse målene for nedside-risiko vil bli brukt i det etterfølgende og hvor ulike resultater de gir vil bli

studert.

Vi vil finne optimale kontrakter for reassurandør, R(X) (se likning 2.2), med hensyn til forskjellige risikomål som minimerer cedentens risiko. Kontraktsformene viser seg ikke å være avhengig av den underliggende fordeling til X. De er derimot forskjellige under forskjellig kriterier. Optimale reforsikringskontrakter under kriteriene ovenfor er studert i for eksempel Cheung, Sung, Yam og Yung (2011), og Chi og Tan (2011) som tar utgangspunkt i lineærkombinasjoner av nedside-mål og forventet gevinst.

Dersom vi anvender vanlig value-at-risk er deres kriterium

$$T_\epsilon = q_\epsilon^{ce} + (1+\nu)E(G^{ce}) \tag{3.5}$$

der $\nu$ er en langrange type parameter. Cheung et al. viser nå at $T_\epsilon$ for en gitt $\nu$ minimeres av reforsikringskontrakter av typen stop-loss

$$X^{re} = \begin{cases} 0 & \text{X} < \text{a} \\ X - a & \text{a} \leq \text{X} \leq \text{a+b} \\ b & \text{X} > \text{a+b} \end{cases}$$

der a og b er to grenser som tilfredsstiller

$$a + b = q_\epsilon^{ce} \quad der \quad Pr(X^{ce} > q_\epsilon^{ce}) = \epsilon \tag{3.6}$$

Dette resultatet gir en oppskrift på hvordan den effisiente Markowitz-grensen mellom forventet gevinst og nedside-risiko kan beregnes. En effisient Markowitz-grense er en kurve som viser forskjellige kombinasjoner av risiko for en gitt forventet gevinst. I vårt tilfelle vil Y-aksen representere risikoen som er uttrykt ved reserven, $q_\epsilon^{ce}$. Mens X-aksen representerer forventet gevinst, $E(G^{ce})$. Selve kurven viser den minimale risiko vi kan oppnå gitt en verdi av forventet gevinst.

Bestem $q_\epsilon$ og minimer $T_\epsilon$ for ulike verdier av $\nu$ med hensyn på koeffisienter a og b som tilfredsstiller (2.8). Dette gir samsvarende verdier av forventet gevinst og nedside-risiko som definerer en kurve av Markowitz-typen. For et forsikringsselskap vil et særlig naturlig kriterium være forholdet

$$F_\epsilon = \frac{E(G^{ce})}{q_\epsilon^{ce}} \tag{3.7}$$

Det kan bevises at $F_\epsilon$ oppnår maksimum når a og b ligger på den efficiente kurven. Det vises hvorfor det oppnår maksimum i delkapittel 3.1.1. Alternativt kan forholdet maksimeres direkte med hensyn på koeffisienter som oppfyller (2.8)

Reassuranse-kontrakten i dette tilfellet kan skrives om til en mer kompakt form

$$\begin{aligned} R(x) &= (x-a)_+ - (x-b^*)_+ \\ &= \mathbf{min}((x-a)_+, b) \end{aligned} \tag{3.8}$$

hvor b* = $a$+$b$.

Som nevnt i delkapittel 2.1 er dette begrenset stop-loss. Denne kontrakten blir også kalt en axb-kontrakt eller forsikringslag. Vi vil i resten av oppgaven referere til denne kontrakten som begrenset stop-loss.

Tilsvarende resultater når value-at-risk $q_\epsilon^{ce}$ byttes ut med betinget value-at-risk $c_\epsilon$ er også etablert, se igjen Cheung et al. (2011). Dersom

$$T_\epsilon = q_\epsilon^{ce} + (1+\nu)E(G^{ce})$$

oppnås minimum når b = $\infty$. Dette gir igjen effisiente Markowitz-grenser mellom forventet gevinst og nedside-risikoen $c_\epsilon$ og maksimering av forholdet

$$F_\epsilon = \frac{E(G^{ce})}{c_\epsilon} \tag{3.9}$$

som omtalt for det andre nedside- kriteriet ovenfor. Når b = $\infty$ kan reassurandør-kontrakten skrives som

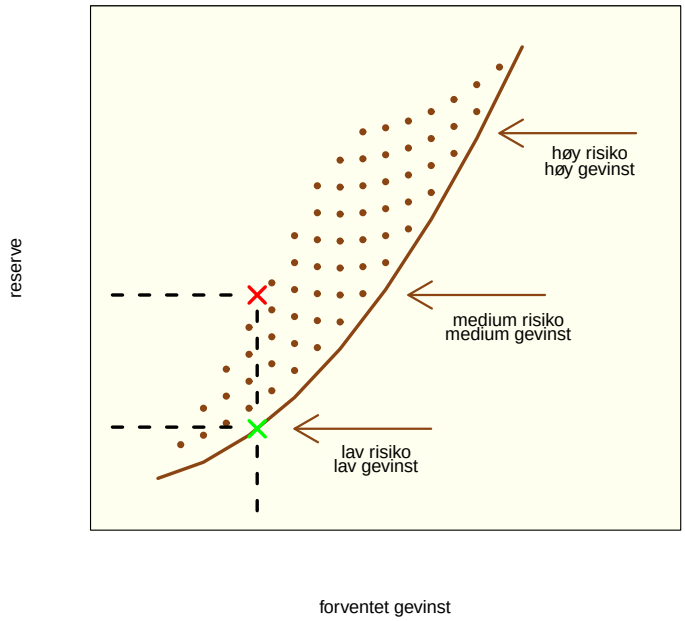
$$R(x) = (x-a)_+ \tag{3.10}$$

Dette er enkel stop-loss-kontrakten som er vist i figur 3 i delkapittel 2.1.

### 3.1.1 Markovitz grense

Markovitz-grensen, på engelsk kalt efficient frontier, er et sentralt begrep innenfor porteføljeteori som bruks til å finne en kombinasjon mellom høyest mulig gevinst og lavest mulig risiko. Harry Markovitz startet med denne ideen i en artikkel i 1952. Han fortsatte å utvikle denne teorien gjennom flere artikler i to tiår. Han vant i 1990 Nobelprisen i økonomi for sine bidrag til porteføljeteori, blant annet gjennom teorien om efficient frontier.

Jeg skal bruke denne teorien for å finne den minste verdien av risikoen, $q_\epsilon^{ce}$ for en gitt verdi av forventet gevinst, $E(G^{ce})$. Oppskriften på Markovitz-grensen er vist i forrige seksjon hvor vi fant at reforsikringskontrakter av typen stop-loss minimerer uttrykket $T_\epsilon$ for en gitt $\nu$. Markovitz-grensen er illustrert i figur 7




Figur 7: *Markovitz-grensen*

Alle punkter over Markovitz-grensen representerer mulige kombinasjoner av risiko og gevinst for en portefølje. Det er ikke mulig å ha kombinasjoner som ligger under kurven. Jo nærmere

grensen du er, jo mindre risiko vil man ha for en gitt gevinst. Eksempelvis er det illustrert to kombinasjoner i figur 7. Det røde krysset representer tilfelle hvor en bedrift har en portefølje med medium risiko for en lav gevinst. Bedriften kan ved hjelp av stop-loss-kontraktene endre kontraktsbetingelsene slik at de får betraktelig lavere risiko for samme verdi av forventet gevinst. Det tilfellet er representert ved hjelp av et grønt kryss. Risikoen reduseres fra medium til lav. Dette betyr dermed at dersom kontraktsparameterene a og b ligger på Markovitz-grensen så vil $F_\epsilon$ bli maksimert siden vi da har oppnåd kombinasjonen som gir minst risiko for en gitt gevinst.

## 3.2 Optimalitet under gitte modeller

Vi skal nå se på optimalitet under forskjellige modeller. Vi tar utgangspunkt i algoritmene 2.1-2.4 fra forrige kapittel. Disse algoritmene simulerte et datasett $X^* = X_1, X_2, \ldots, X_m$ hvor

$$\mathrm{N}_j = \mathrm{poisson}(\lambda)$$

$$Z_{ij}|N_j \sim \text{sannsynlighetsmodell}(\alpha, \beta), \quad i = 1, \ldots, N_j \tag{3.11}$$

$$X_j = \sum_{i=1}^{N_j} Z_i, \qquad \text{j=1,. . . ,m}$$

Algoritmene fra forrige kapittel repeterte dette $m$ ganger og returnerte $m$ replikasjoner av aggregerte krav $X^* = X_1, \ldots, X_m$. Slik simulerer vi datasettet vi skal ta utgangspunkt i for å kalkulere optimal reassuranse. Kravstørrelsesen Z setter jeg foreløpig til å følge en vilkårlig sannsynlighetsmodell.

Vi har i forrige delkapittel vist to optimale kontrakter for R(X). Det gjenstår å finne estimater på grensene a og b for begrenset stop-loss og $a_e$ for enkel stop-loss som optimerer hendholdsvis kriterie 3.7 og 3.9. Jeg vil heretter anta at vi har tilgang til et datasett

$$X_k^* = (p(\alpha_k, \beta_k), \lambda, m), \quad k = 1, \ldots, 9 \tag{3.12}$$

som har en underliggende sannsynlighetsfordeling, $p$, for kravstørrelsene $Z_i$ med parametre $\alpha_k$ og $\beta_k$. Jeg skal se på optimalitet for 9 forskjellige parametersett for hver fordeling, ergo k=1,. . .,9. $\lambda = J\mu\mathrm{T}$ er forventet antall krav per portefølje over en tidsperiode T, antall poliser J og kravintensitet $\mu$ per polise per tidenhet T. $m$ er antall replikasjoner.

**Simulering**
Jeg definerer to funksjoner $f_1$ og $f_2$ som returnerer henholdsvis kriteriene 3.7 og 3.9.

**Funksjon 3.1** Beregning av value-at-risk kriteriet

```
1   f1=function(a,b,X){
2
3     R(X)=min((x − a)+, b)        #Reassurandør
4     C(X)=X-min((x − a)+, b)      #Cedent
5     π=E(X)
6     π^re=E(R(X))
7     E(G)=γπ − γ^re π^re
8     q_ϵ^ce=sort(C(X))[ϵ*m]
9
10  return E(G)/q_ϵ^ce
11  }
```

**Funksjon 3.2** Beregning av betinget value-at-risk kriteriet

```
1   f2=function(a,X){
2
3     R(X)=(x − a)+          #Reassurandør
4     C(X)=X-(x-a)+          #Cedent
5     π=E(X)
6     π^re=E(R(X))
7     E(G)=γπ − γ^re π^re
8     q_ϵ^ce=sort(C(X))[ϵ*m]
9     c_ϵ^ce=E[C(X)>q_ϵ^ce]
10
11  return E(G)/c_ϵ^ce
12  }
```

Disse to funksjonene brukes til å finne estimerte verdier av grensene a, b og $a_e$ i en enkel stop-loss-kontrakt og en begrenset stop-loss-kontrakt. Dette gjøres ved hjelp av optimeringsfunksjonene *optim*[2] og *optimize*[3] som man kan se i algoritme 3.3. Disse funksjonene tar inn et datasett og en funksjon som skal optimeres. De returner grenseverdier som optimerer den gjeldende funksjonen på det gitte datasettet. Funksjonen *optimize* bruker en kombinasjon av *Golden Section Search* og *Successive Parabolic interpolation*. *Optim* har flere metoder innebygd. *Neldor* og *Mead* metoden var den metoden som ga mest stabilitet. Dette er metoden som er satt som standard i R. Med stabilitet mener jeg stadig konvergens av estimatene.

[2]optim er en innebygd funksjon i R som optimerer flere variabler med hensyn til en gitt funksjon og et datasett

[3]optimize er en innebygd funksjon i R som optimerer èn variabel med hensyn til en gitt funksjon og et datasett

**Algoritme 3.3** Reassuranse-optimering

```
1  Inn: X, f1, f2
2
3  [â,b̂] = optim(f1,data=X)          #Begrenset stop-loss
4  [a_e] = optimize(f2,data=X)       #Enkel stop-loss
```

Optimeringen av funksjonene går som regel problemfritt, men noen sjeldne ganger kan man få problemer med konvergens for parameteren $b$. De gangene optimeringen feilet fikk vi et estimat på $b$ som var den maksimale verdien i datasettet X. De andre metodene i *optim* var meget ustabile, mens standardmetoden konvergerte fint for det meste. For å gjøre disse beregningene så stabile som mulig er det fint å ta utgangspunkt i fornuftige startverdier. Fornuftige startverdier for $a$ og $b$ kan være henholdsvis $E(X)$ og $q_\epsilon - E(X)$ som man i lys av tabellene 3.1-3.4 kan forstå er rimelig. Disse startverdiene er et resultat av litt erfaring med beregningene og kriteriet (likning 3.6) som ble indrodusert i forrige seksjon. Dette kriteriet er også en fin måte å sjekke på om estimatene stemmer da det må være oppfylt.

I de neste seksjonene skal vi se på hvordan de optimale verdiene av $\hat{a}$, $\hat{b}$ og $\hat{a}_e$ endrer seg når man endrer parametrene til den underliggende fordelingen, $p(\alpha_k, \beta_k)$, som er størrelsen på kravene Z. Parameteren til enkel stop-loss definerer vi som $a_e$ for å skille mellom kontraktparametrene. Skal se på 9 forskjellige parametersett

$$(\alpha_k, \beta_k), \quad k = 1, \ldots, 9$$

for alle fire type modellene som er vist i kapittel 2.2. 9 forskjellige parametersett vil gi oss 9 forskjellige datasett

$$(X_1^*, X_2^*, X_3^*, X_4^*, X_5^*, X_6^*, X_7^*, X_8^*, X_9^*)$$

for hver type modell. For hver av disse datasettene skal vi beregne $a$, $b$, $a_e$, $\mathrm{E}(X_{kj}^*)$, $\mathrm{SD}(X_{kj}^*)$ og tilhørende optimale $F_\epsilon$-verdi. Ladningene i funksjonene f1 og f2 setter jeg til å være 0.1 og 0.2 for henholdsvis $\gamma$ og $\gamma^{re}$ i resten av avhandlingen. Ladningene brukes i likning 3.2 for beregning av forventet gevinst.

### 3.2.1 Poisson/lognormal

Den overnevnte simuleringen ble gjennomført med disse tallene

$$\underset{\textit{antall poliser}}{\mathrm{J} = 1000} \qquad \underset{\textit{krav intensitet}}{\mu = 10\%} \qquad \underset{\textit{antall replikasjoner}}{\mathrm{m} = 100000} \qquad \underset{\textit{tidsenhet}}{\mathrm{T} = 1}$$

$$\mathrm{N}^* \sim \text{poisson}(\mathrm{J}\mu\mathrm{T})$$
$$\Downarrow$$
$$Z_k^* \sim p(\alpha_k, \beta_k) = \textit{log-normal}(\alpha_k, \beta_k), \quad \mathrm{k}=1,\ldots,9$$
$$\Downarrow$$
$$(\mathrm{X}_1^*, X_2^*, X_3^*, X_4^*, X_5^*, X_6^*, X_7^*, X_8^*, X_9^*)$$

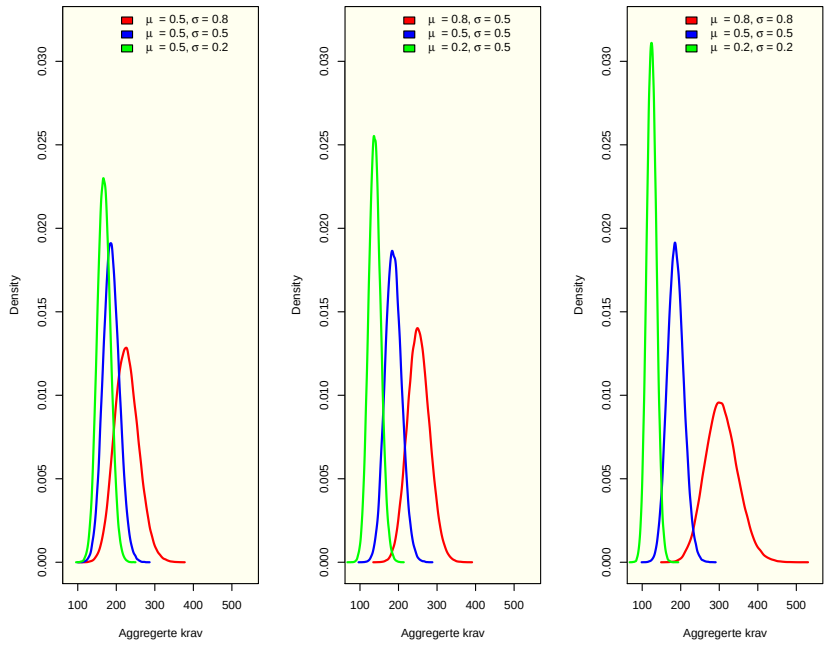
Som vist over, er størrelsen på kravene i denne seksjonen lognormal-fordelt med parameter $\alpha$ og $\beta$. I lognormal-tilfellet er $\alpha$ forventningen og $\beta$ standardavviket. I tabell 3.1 er de forskjellige parameterkombinasjonene listet opp. Det er 9 forskjellige parameterkombinasjoner. Disse gir oss et tilsvarende datasett. For hver av datasettene har jeg estimert kontraktgrensene a, b og

$a_e$ som optimerer forventet gevinst mot nedsiderisiko. Resultatene er vist i tabellen. Femte og sjette kolonne i tabellen er forventingen og standardavviket til de respektive datasettene. Siste kolonne er kriteriet (likning 3.7) knyttet til begrenset stop-loss hvor man tar hensyn til vanlig value-at-risk.

| **Parametre** | **Data** | **Begrenset stop loss** | **Enkel stop loss** | $\mathbf{E}(X_k^*)$ | $\mathbf{Sd}(X_k^*)$ | $\mathbf{F}\epsilon$ |
|---|---|---|---|---|---|---|
| $\alpha = 0.5, \beta = 0.8$ | $X_{1j}^*$ | a = 230 b = 76 | $a_e = 230$ | 227 | 31 | 0.0892 |
| $\alpha = 0.5, \beta = 0.5$ | $X_{2j}^*$ | a = 189 b = 49 | $a_e = 189$ | 187 | 21 | 0.0910 |
| $\alpha = 0.5, \beta = 0.2$ | $X_{3j}^*$ | a = 170 b = 40 | $a_e = 170$ | 168 | 17 | 0.0920 |
| $\alpha = 0.8, \beta = 0.5$ | $X_{4j}^*$ | a = 255 b = 67 | $a_e = 255$ | 252 | 29 | 0.0910 |
| $\alpha = 0.5, \beta = 0.5$ | $X_{5j}^*$ | a = 189 b = 49 | $a_e = 189$ | 187 | 21 | 0.0911 |
| $\alpha = 0.2, \beta = 0.5$ | $X_{6j}^*$ | a = 140 b = 37 | $a_e = 140$ | 138 | 16 | 0.0910 |
| $\alpha = 0.8, \beta = 0.8$ | $X_{7j}^*$ | a = 312 b = 101 | $a_e = 310$ | 306 | 42 | 0.0892 |
| $\alpha = 0.5, \beta = 0.5$ | $X_{8j}^*$ | a = 189 b = 49 | $a_e = 189$ | 187 | 21 | 0.0910 |
| $\alpha = 0.2, \beta = 0.2$ | $X_{9j}^*$ | a = 125 b = 30 | $a_e = 126$ | 125 | 13 | 0.0920 |

Tabell 3.1: *Optimal reassuranse for lognormal-fordelte krav*

Siden kontraktgrensene har en direkte sammenheng med reserven for det gitte datasettet, likning 3.6, kan vi forvente å se høyere grenseverdier for datasettene med høyere forventning og standardavvik. Det kan vi se i tabell 3.1. De tre første radene i tabellen reduserer jeg $\beta$ gradvis. Det betyr at spredningen på kravstørrelsene reduseres. Det fører til at tilsvarende datasett, $X_k^*$, får lavere forventning og standardavvik. Vi ser dermed at de tilsvarende grenseverdiene blir mindre. Nedre grense for reassurandør, a, reduseres fra 196 til 151. Intervallet b, hvor reassurandør tar over all risiko går fra å være 106 til 57. Fra rad 4 til 6 reduserer jeg $\alpha$, forventningen til kravstørrelsene. Vi ser også her at tilsvarende verdier for a og b reduseres. I rad 7 til 9 i tabellen ser vi en kraftigere reduksjon for grenseverdiene a og b. Det kommer av at jeg reduserer både $\alpha$ og $\beta$. I figur 11 kan vi se hvordan fordelingen til datasettet for de aggregerte kravene forandrer seg med forskjellige paramaterverdier i fordelingen til kravstørrelsene.




Figur 8: *Fordelingen til totalkravene $X_k^*$ for gitte parameterkombinasjon med lognormale-fordelte krav*

### 3.2.2 Poisson/gamma

Bruker samme prosedyre i denne seksjonen, men størrelsen på kravene i denne seksjonen er gamma-fordelt.

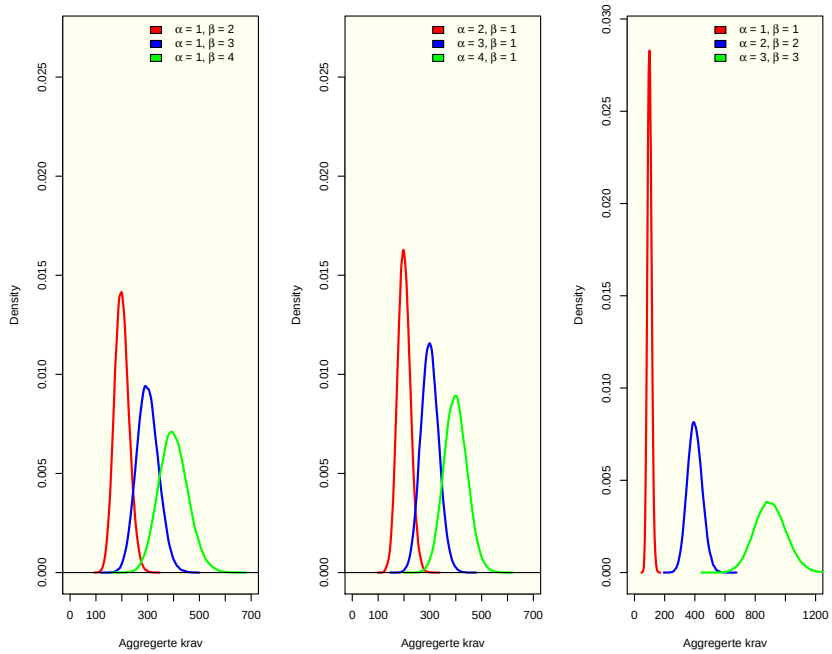
$$\underset{\textit{antall poliser}}{\mathrm{J}=1000} \qquad \underset{\textit{krav intensitet}}{\mu=10\%} \qquad \underset{\textit{antall replikasjoner}}{\mathrm{m}=100000} \qquad \underset{\textit{tidsenhet}}{\mathrm{T}=1}$$

$$\mathrm{N}^* \sim \text{ poisson}(\mathrm{J}\mu\mathrm{T})$$
$$\Downarrow$$
$$Z_k^* \sim p(\alpha_k, \beta_k)=\text{Gamma}(\alpha_k, \beta_k), \quad \text{k=1,...,9}$$
$$\Downarrow$$
$$(\mathrm{X}_1^*, X_2^*, X_3^*, X_4^*, X_5^*, X_6^*, X_7^*, X_8^*, X_9^*)$$

| **Parametre** | **Data** | **Begrenset stop loss** | **Enkel stop loss** | $\mathbf{E}(X_k^*)$ | $\mathbf{Sd}(X_k^*)$ | $\mathbf{F}\epsilon$ |
|---|---|---|---|---|---|---|
| $\alpha=1,\beta=2$ | $X_{1j}^*$ | a = 203 b = 68 | $a_e = 203$ | 200 | 28 | 0.0889 |
| $\alpha=1,\beta=3$ | $X_{2j}^*$ | a = 304 b = 101 | $a_e = 304$ | 300 | 42 | 0.0888 |
| $\alpha=1,\beta=4$ | $X_{3j}^*$ | a = 406 b = 133 | $a_e = 406$ | 399 | 56 | 0.0886 |
| $\alpha=2,\beta=1$ | $X_{4j}^*$ | a = 202 b = 57 | $a_e = 202$ | 200 | 24 | 0.0903 |
| $\alpha=3,\beta=1$ | $X_{5j}^*$ | a = 303 b = 81 | $a_e = 303$ | 300 | 35 | 0.0909 |
| $\alpha=4,\beta=1$ | $X_{6j}^*$ | a = 404 b = 105 | $a_e = 404$ | 400 | 45 | 0.0912 |
| $\alpha=1,\beta=1$ | $X_{7j}^*$ | a = 102 b = 33 | $a_e = 102$ | 100 | 14 | 0.0887 |
| $\alpha=2,\beta=2$ | $X_{8j}^*$ | a = 405 b = 117 | $a_e = 405$ | 400 | 49 | 0.0904 |
| $\alpha=3,\beta=3$ | $X_{9j}^*$ | a = 909 b = 244 | $a_e = 909$ | 900 | 104 | 0.0909 |

Tabell 3.2: *Optimal reassuranse for gamma-fordelte krav*

Med gamma-fordelte kravstørrelser vises det resultater for forskjellige $\beta$-verdier for de 3 første radene i tabellen. Fra rad 4-6 vises det resultater for forskjellige $\alpha$-verdier. Vi ser at nedre grense for reassurandør, a, er nærmest likt for datasett 1 og 4, 2 og 5, 3 og 6. Det kommer av at forventningen de tilsvarende datasettene også er de samme. Ser vi derimot på intervallet b ser man at tilsvarende økninger i $\alpha$ og $\beta$ fører til forskjellige lengder på intervallet. Det kan kobles opp mot standardavvikene til datasettene som blir forskjellige for tilsvarende økninger for $\alpha$ og $\beta$. Det kommer av egenskapene i gamma-fordelingen. Det er verdt å merke seg resultatene fra rad 3 og 6. Datasett 3 og 6 har samme forventning, noe som fører til relativt like grenseverdier for a. Men standardavviket er større for datasett 3. Det fører dermed til at intervallet b blir en del lengre for datasett 3 i forhold til 6 selv om forventningene er ganske like. Det ser ut til at jo større spredning det er, jo større blir ansvaret til reassurandør.




Figur 9: *Fordelingen til totalkravene $X_k^*$ for gitte parameterkombinasjon med gamma-fordelte krav*

Ser vi på de siste 3 parameterkombinasjonene ser vi en stor økning i forventingen og standardavviket til det underliggende datasettet, noe som kommer tydelig frem i plot 3 fra venstre i figur 9. Når vi øker begge parametrene vil forventingen og variansen øke drastisk. I praksis betyr det at disse parameterkombinasjonene gir oss fordelinger med høyere sannsynligheter for større krav i tillegg til et større spekter av kravstørrelser. Vi kan se fra tabell 3.2 at de optimale verdiene a og b får høye verdier. Det betyr at den nedre grensen for reassurandør, a, blir høyere, men det blir også intervallet b, hvor cedenten bare står til ansvar for en fast sum b, mens reassurandør tar på seg den øvrige risikoen. Grensen a for enkel stop-loss har i dette tilfellet også tilnærmet lik verdi som a i begrenset stop-loss.

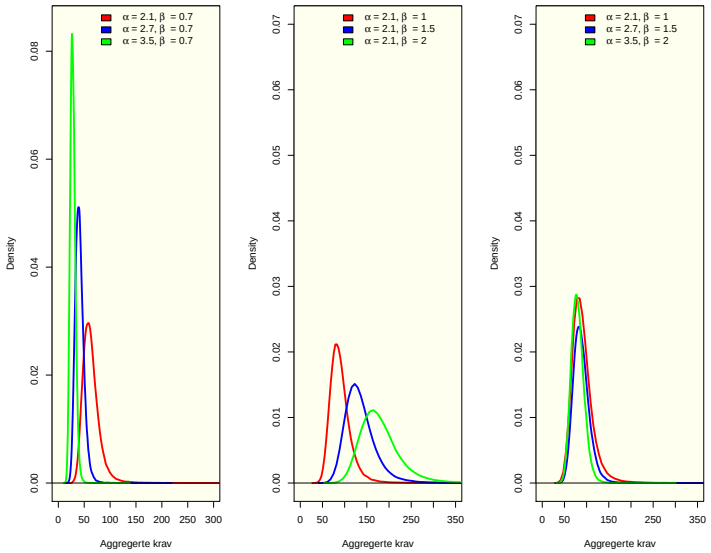
### 3.2.3 Poisson/pareto

Jeg brukte det samme oppsettet som i forrige seksjon, men kravstørrelsene er i denne seksjonen pareto-fordelt.

$$\underset{\textit{antall poliser}}{\mathrm{J} = 1000} \qquad \underset{\textit{krav intensitet}}{\mu = 10\%} \qquad \underset{\textit{antall replikasjoner}}{\mathrm{m} = 100000} \qquad \underset{\textit{tidsenhet}}{\mathrm{T} = 1}$$

$$
\begin{gathered}
\mathrm{N}^{*} \sim \text{ poisson}(\mathrm{J}\mu\mathrm{T}) \\
\Downarrow \\
Z_k^* \sim p(\alpha_k, \beta_k) = \text{Pareto}(\alpha_k, \beta_k), \quad \mathrm{k}=1,\ldots,9 \\
\Downarrow \\
(\mathrm{X}_1^*, X_2^*, X_3^*, X_4^*, X_5^*, X_6^*, X_7^*, X_8^*, X_9^*)
\end{gathered}
$$

Vi kan først ta en titt på plot 1 i figur 10 hvor vi kan se sannsynlighetsfordelingene til $X_k^*$ med forskjellige $\alpha$-verdier. Fordelingene er veldig spisse, noe som vitner om små verdier for standardavviket. Vi ser at jo større $\alpha$-verdien er i dette tilfellet, jo mindre vil standardavviket bli. Det samme blir forventingen. Dette vil resultere i mindre a-, b- og a (enkel stop-loss)-verdier. I tabell 3.3 kan vi se at standardavviket faller drastisk mens vi vitner til moderate fall i forventingen. Dette betyr med erfaring fra de forrige seksjonene at intervallet b vil bli drastisk mindre for fall i $\alpha$-verdi men a og $a_e$ vil vise moderate fall i takt med forventingen.




Figur 10: *Fordelingen til totalkravene $X_k^*$ for gitte parameterkombinasjon med pareto-fordelte krav*

Når vi ser på forskjellige $\beta$-verdier ser vi fra plot 2 i figur 10 at fordelingene fremtrer med motsatt effekt enn en økning i $\alpha$. Både forventingen og standardavviket øker når $\beta$ øker. Vi får porteføljer som representerer krav for større størrelser og med større spredning. Dette vil naturligvis resultere i høyere verdier for alle grensene for begge reassuranse-kontraktene.

| **Parametere** | **Data** | **Begrenset stop-loss** | **Enkel stop-loss** | $\mathbf{E}(X_k^*)$ | $\mathbf{Sd}(X_k^*)$ | $\mathbf{F}\epsilon$ |
|---|---|---|---|---|---|---|
| $\alpha = 2.1, \beta = 0.7$ | $X_{1j}^*$ | a = 64 b = 61 | $a_e = 64$ | 64 | 20 | 0.0816 |
| $\alpha = 2.7, \beta = 0.7$ | $X_{2j}^*$ | a = 42 b = 25 | $a_e = 42$ | 41 | 9 | 0.0844 |
| $\alpha = 3.5, \beta = 0.7$ | $X_{3j}^*$ | a = 29 b = 13 | $a_e = 29$ | 28 | 5 | 0.0857 |
| $\alpha = 2.1, \beta = 1.0$ | $X_{4j}^*$ | a = 92 b = 88 | $a_e = 92$ | 91 | 31 | 0.0815 |
| $\alpha = 2.1, \beta = 1.5$ | $X_{5j}^*$ | a = 137 b = 128 | $a_e = 138$ | 136 | 45 | 0.0807 |
| $\alpha = 2.1, \beta = 2.0$ | $X_{6j}^*$ | a = 183 b = 172 | $a_e = 184$ | 182 | 58 | 0.0804 |
| $\alpha = 2.1, \beta = 1.0$ | $X_{7j}^*$ | a = 91 b = 88 | $a_e = 92$ | 91 | 37 | 0.0815 |
| $\alpha = 2.7, \beta = 1.5$ | $X_{8j}^*$ | a = 90 b = 54 | $a_e = 90$ | 88 | 20 | 0.0845 |
| $\alpha = 2.5, \beta = 2.0$ | $X_{9j}^*$ | a = 81 b = 39 | $a_e = 81$ | 80 | 15 | 0.0861 |

Tabell 3.3: *Optimal reassuranse for pareto-fordelte krav*

Fra rad 7 til 9 i tabell 3.3 øker vi verdiene for både $\alpha$ og $\beta$. Ved hjelp av figur 10 er det vanskelig å se hvor forskjellene ligger. Fra tabellen ser vi at det er ikke stor forskjell på forventningen. Det er i standardavviket forskjellene ligger. Standardavviket minker når $\alpha$ og $\beta$ øker i verdi og det samme gjør verdiene for b, som forventet. For de optimale verdiene av a og $a_e$ er det ikke nevneverdige forskjeller.

### 3.2.4 Stokastisk intensitet

**Resultater**
Her blir det interessant å se hvordan $\psi$ vil på virke reassuranse-kontraktene. Oppsettet er som følger

$$\underset{\textit{antall poliser}}{\mathrm{J} = 1000} \qquad \underset{\textit{forventet intensitet}}{\xi{=}0.1} \qquad \underset{\textit{antall replikasjoner}}{\mathrm{m} = 100000} \qquad \underset{\textit{tidsenhet}}{\mathrm{T} = 1}$$

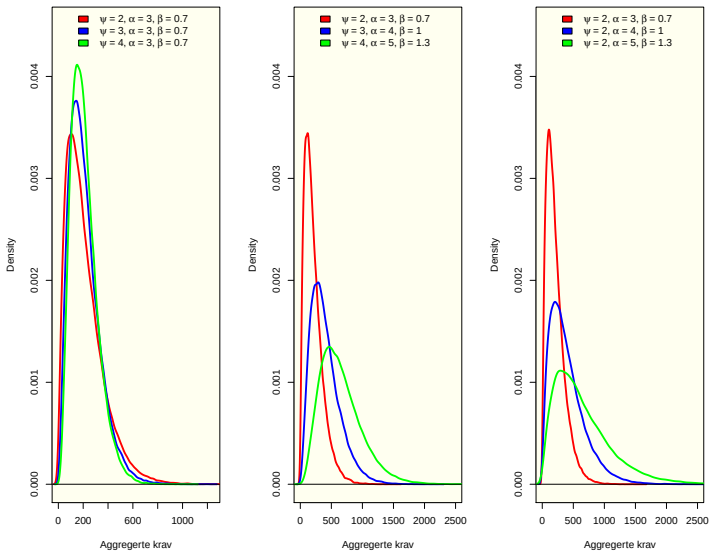
$$\mathrm{N}^* \sim \text{poisson}(\mathrm{J}\mu\mathrm{T})$$
$$\Downarrow$$
$$Z_k^* \sim p(\alpha_k, \beta_k, \xi) = \text{Gamma\_stokastisk}(\alpha_k, \beta_k, \xi), \quad \mathrm{k}=1,\ldots,9$$
$$\Downarrow$$
$$(X_1^*, X_2^*, X_3^*, X_4^*, X_5^*, X_6^*, X_7^*, X_8^*, X_9^*)$$

| Parametere | Data | Begrenset stop-loss | Enkel stop-loss | $\mathbf{E}(X_k^*)$ | $\mathbf{Sd}(X_k^*)$ | $\mathbf{F}\epsilon$ |
|---|---|---|---|---|---|---|
| $\psi$=2,$\alpha = 3, \beta = 0.7$ | $X_{1j}^*$ | a = 280 b = 423 | $a_e = 283$ | 210 | 150 | 0.0500 |
| $\psi$=3,$\alpha = 3, \beta = 0.7$ | $X_{2j}^*$ | a = 257 b = 338 | $a_e = 259$ | 210 | 124 | 0.0571 |
| $\psi$=4,$\alpha = 3, \beta = 0.7$ | $X_{3j}^*$ | a = 246 b = 291 | $a_e = 247$ | 210 | 108 | 0.0623 |
| $\psi$=2,$\alpha = 3, \beta = 0.7$ | $X_{4j}^*$ | a = 281 b = 418 | $a_e = 283$ | 210 | 150 | 0.0509 |
| $\psi$=2,$\alpha = 4, \beta = 1.0$ | $X_{5j}^*$ | a = 535 b = 812 | $a_e = 540$ | 401 | 288 | 0.0571 |
| $\psi$=2,$\alpha = 5, \beta = 1.3$ | $X_{6j}^*$ | a = 868 b = 1298 | $a_e = 874$ | 650 | 463 | 0.0611 |
| $\psi$=2,$\alpha = 3, \beta = 0.7$ | $X_{7j}^*$ | a = 281 b = 426 | $a_e = 283$ | 211 | 151 | 0.0511 |
| $\psi$=3,$\alpha = 4, \beta = 1.0$ | $X_{8j}^*$ | a = 492 b = 643 | $a_e = 495$ | 401 | 235 | 0.0370 |
| $\psi$=4,$\alpha = 5, \beta = 1.3$ | $X_{9j}^*$ | a = 763 b = 894 | $a_e = 767$ | 651 | 334 | 0.0316 |

Tabell 3.4: *Optimal reassuranse med stokastisk intensitet*

Ser vi på de 3 første radene i tabell 3.4 så simulerte jeg datasett med forskjellige $\psi$-verdier mens jeg holdt de andre to parametrene fast. Forventningen til de tilsvarende datasettene forblir det samme for endringer i $\psi$, men vi kan se en merkbar effekt på standardavviket. Det blir mindre for større verdier av $\psi$. Dette stemmer overens med egenskapene til en standard gamma-modell vi viste i seksjon 2.2.4. Vi kan se at $\psi$ er i nevneren i likning 3.4 og større verdier av parameteren vil naturligvis redusere standardavviket i størrelse. Dette gir utslag i plottet til venstre i figur 11 hvor vi ser fordelingene bli smalere og spissere. Som allerede nevnt i tidligere seksjoner vil det påvirke lengden på intervallet b, hvor cedenten står til ansvar for en fast sum. Fra tabell 3.4 ser vi en kraftig reduksjon for intervallet b i takt med at parameteren $\psi$ blir mindre. Som forventet viser ikke grensene a og $a_e$ store endringer.

For de neste 3 parameterkombinasjonene, rad 4-6, økte jeg både $\alpha$ og $\beta$ mens $\psi$ var fastsatt. Her ser vi en kraftig økning i både forventningen og standardavviket til de tilsvarende datasettene. Det resulterer dermed i økte grenseverdier for a og b. Det samme gjelder for a i en enkel stop-loss.




Figur 11: *Fordelingen til totalkravene $X_k^*$ for gitte parameterkombinasjon*

Til slutt så jeg på tilfellet hvor alle parametrene øker i verdi, rad 7-9. Her ser vi at økninger i alle parametrene fører til økning i standardavviket til det tilsvarende datasettet. Datasettene fra rad 1-3 viste at standardavviket ble redusert for økninger i parameteren $\psi$. Dette betyr at effekten av økninger i $\alpha$ og $\beta$ overstyrer effekten av økninger i $\psi$. Det vil igjen bety at vi får større verdier for b for økninger i alle parametrene. Det kan vi se fra tabell 3.4. Det samme gjør nedre grense for reassurandør, a, siden forventningen til de tilsvarende datasettene øker.

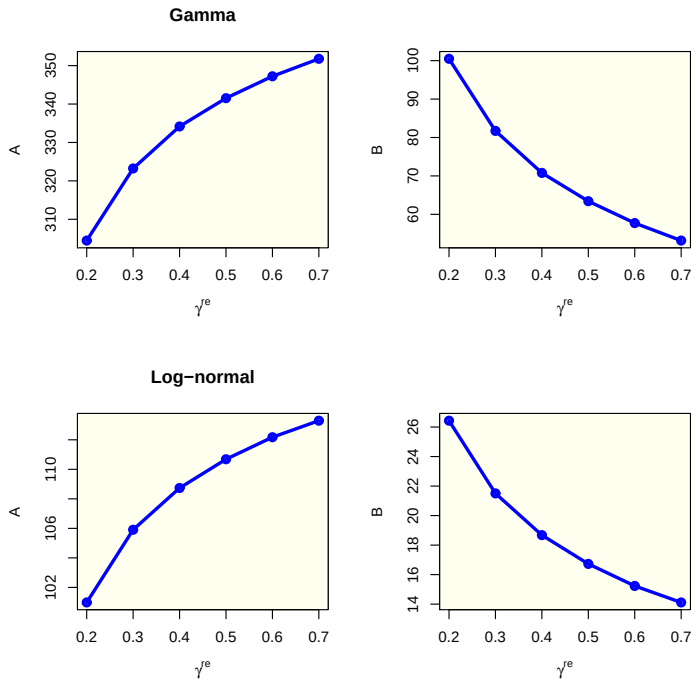
## 3.3 Optimalitet for gitte reassuranse-priser og solvensnivå

Når cedenten overfører deler av risikoen over til reassurandør vil de ta et prispåslag. $(1+\gamma^{re})$ er prispåslaget hvor ladningen $\gamma^{re}$ varierer ut ifra markedstilstandene. Dette er definert i likning 3.2 i starten av kapittelet. I det foregående kapittelet (3.2) satt vi en fast verdi for ladningen. $\gamma^{re}$=0.2. Det vil være interessant å se hvordan de optimale grensene endrer seg med forskjellige verdier for ladningen. Det er jo naturlig å tenke at disse prisene vil variere med markedstilstanden og cedenten må derfor ta en vurdering utifra disse.

| **m=100000** | **A=100000** | | **Gamma** $\rightarrow \alpha = 1, \beta = 3$ | | | **Lognormal** $\rightarrow \alpha = 1, \beta = 0.5$ |
|---|---|---|---|---|---|---|
| | **Gamma** | | | **Lognormal** | | |
| | a | b | $F_\epsilon^o$ | a | b | $F_\epsilon^o$ |
| $\gamma^{re}$=0.2 | 305 | 100 | 0.08888 | 101 | 26 | 0.09114 |
| $\gamma^{re}$=0.3 | 323 | 82 | 0.08547 | 106 | 21 | 0.08834 |
| $\gamma^{re}$=0.4 | 334 | 71 | 0.08352 | 109 | 18 | 0.08671 |
| $\gamma^{re}$=0.5 | 342 | 63 | 0.08220 | 111 | 16 | 0.08560 |
| $\gamma^{re}$=0.6 | 347 | 58 | 0.08121 | 112 | 15 | 0.08477 |
| $\gamma^{re}$=0.7 | 352 | 53 | 0.08043 | 113 | 14 | 0.08411 |

Tabell 3.5: *Optimal reassuranse for gitte priser*

Resultatene i tabell 3.5 viser beregnet optimal reassuranse med begrenset stop-loss for gitte reassuranse-ladninger. Jeg ser på seks forskjellige verdier av ladningen $\gamma^{re}$. Tabellen viser med andre ord hvordan grensene til en begrenset stop-loss-kontrakt vil endre seg når kjøp av reassuranse blir dyrere. Grensen $b$ og forholdstallet $F_\epsilon^o$ blir mindre for høyere verdier av ladningen mens grensen $a$ øker. Hva betyr dette? Lavere $b$ betyr at intervallet hvor resssurandør tar over all risiko blir kortere. Det betyr at cedenten kutter ned på andelen av reassuranse jo høyere ladningen blir. Samtidig vet vi at a+b=$q_\epsilon$ alltid er oppfylt. Dette betyr at $a$ øker og cedenten tar over en større andel av risikoen.



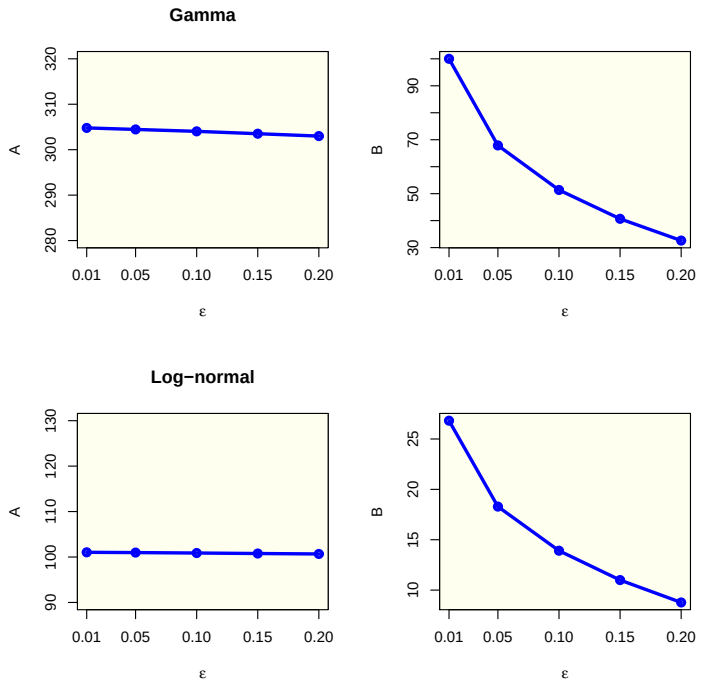

Figur 12: *Optimale grenser under gitte priser*

I figur 13 er verdiene for grensene i begrenset stop-loss-kontrakten illustrert for forskjellige verdier av $\gamma^{re}$. Vi ser som nevnt over at $a$ øker i verdi og $b$ reduseres i verdi. Det ser ikke ut som det er nevneverdige forskjeller mellom disse to modellene da begge verdiene beveger seg ganske likt. Det vil også være interessant å se hvordan grensene endrer seg med $\epsilon$. Det er viktig å huske at betingelsen a+b=$q_\epsilon$ alltid må være oppfylt. Høyere verdier av $\epsilon$ betyr at sannsynligheten for value-at-risk blir høyere. Det frigjøres kapital og dermed får vi mindre avsatt kapital for eventuelle ekstreme kravstørrelser. Vi antar i det etterfølgende at $\gamma^{re}$ er satt til 0.2.

| **m=100000** | **A=100000** | | **Gamma** $\rightarrow \alpha = 1, \beta = 3$ | | **Lognormal** $\rightarrow \alpha = 1, \beta = 0.5$ | |
|---|---|---|---|---|---|---|
| | **Gamma** | | | **Lognormal** | | |
| | a | b | $F_\epsilon^o$ | a | b | $F_\epsilon^o$ |
| $\epsilon$=0.01 | 305 | 100 | 0.08885 | 101 | 27 | 0.09104 |
| $\epsilon$=0.05 | 305 | 67 | 0.08940 | 101 | 18 | 0.09148 |
| $\epsilon$=0.10 | 304 | 51 | 0.09021 | 101 | 14 | 0.09212 |
| $\epsilon$=0.15 | 304 | 40 | 0.09112 | 101 | 11 | 0.09287 |
| $\epsilon$=0.20 | 303 | 33 | 0.09211 | 101 | 8 | 0.09367 |

Tabell 3.6: *Optimal reassuranse for gitt solvensnivå*

Vi ser i dette tilfellet, fra tabell 3.6, en redusering for intervallet $b$ i takt med økning i $\epsilon$. Grensen $a$ holder seg konstant i dette tilfellet for å oppnå den beste kombinasjonen av forventet gevinst og risiko. Det resulterer i at forholdstallet $F_\epsilon^o$ øker for større verdier av $\epsilon$. Verdiene for $a$ og $b$ er illustrert i figur 13.




Figur 13: *Optimale grenser under gitte priser*

## 3.4 Sammenlikning med proporsjonal reassuranse

Proporsjonal reassuranse er en kontrakt hvor en fast prosentandel, c, av det totale kravet utgjør reassurandørs ansvar. I vårt tilfelle snakker vi om aggregerte krav X.

$$R(X) = X^{re} = cX, \qquad X^{ce} = X - X^{re} \tag{3.13}$$

hvor c $\in$ [0,1]. Et optimalt forholdstall (likning 3.7) for en proporsjonal reassuranse-kontrakt tilsvarer det samme som å ikke ha en kontrakt i det hele tatt. En fast prosent andel av kravet skal cedenten betale uansett størrelse på kravet.

Man vil med dette sammenlikne det optimale forholdstallet for en proporsjonal-kontrakt mot en begrenset stop-loss-kontrakt. Jeg vil gjøre dette for hver type portefølje vi har sett i dette kapittelet.

| Kontrakt | Estimerte parametre | Forholdstall | Forskjell i % |
|---|---|---|---|
| **Normal** | | | |
| Proposjonal | c = 4.59e-05 | F = 0.0783 | |
| Stop loss | a = 311 b = 82 | F = 0.0911 | 14.1 |
| **Gamma** | | | |
| Proposjonal | c = 4.59e-05 | F = 0.0741 | |
| Stop loss | a = 305 b = 100 | F = 0.0889 | 16.6 |
| **Pareto** | | | |
| Proposjonal | c = 4.59e-05 | F = 0.0487 | |
| Stop loss | a = 100 b = 105 | F = 0.0807 | 39.7 |
| **Stokastisk** | | | |
| Proposjonal | c = 4.59e-05 | F = 0.0148 | |
| Stop loss | a = 533 b = 1140 | F = 0.0261 | 43.1 |

Tabell 3.7: *Optimal proporsjonal- og begrenset stop-loss-kontrakt*

Vi ser fra tabell 3.5 at optimering av forholdstallet for en proporsjonal-kontrakt gir oss en parameterverdi for $c$ som er veldig nær 0. Det betyr at den største delen av risikoen ligger hos cedenten når forholdstallet er optimert. Siste rad i tabellen er forholdet mellom forholdstallene mellom en proporsjonal-kontrakt og begrenset stop-loss-kontrakt oppgitt i prosent for hver type portefølje. Kontraktene evalueres på samme datasett og vi ser at forskjellene mellom forholdstallene er størst for den stokastiske porteføljen. En begrenset stop-loss-kontrakt gir et forholdstall som er 43.1 % bedre enn en proporsjonal-kontrakt. Vi ser at en begrenset stop-loss-kontrakt gir oss et bedre forholdstall uansett modell.

| Persentil | 50% | 80% | 99% |
|---|---|---|---|
| Normal | | | |
| Uten reassuranse | 307.07 | 336.91 | 392.62 |
| Proporsjonal reassuranse | 307.05 | 336.89 | 392.60 |
| Optimal reassuranse | 307.07 | 310.87 | 310.87 |
| Gamma | | | |
| Uten reassuranse | 298.72 | 335.52 | 405.57 |
| Proporsjonal reassuranse | 298.71 | 335.50 | 405.55 |
| Optimal reassuranse | 298.72 | 304.51 | 304.51 |

Tabell 3.8: *Cedentens persentil av totalskade*

Som forventet ser vi at proporsjonal reassuranse ikke påvirker cedentens kapitalreserver i forhold til situasjonen hvor man ikke har reassuranse. Cedentens kapitalreserve får en signifikant endring med optimal reassuranse. Vi ser at det må avsettes 20% mindre kapital med en optimal kontrakt hvor skadene er normal-fordelte og 25% mindre hvor skadene er gamma-fordelte. Dette er store prosentverdier som vil gi et signifikant utslag på mengden av penger som frigjøres.

# 4 Optimal reassuranse i praksis

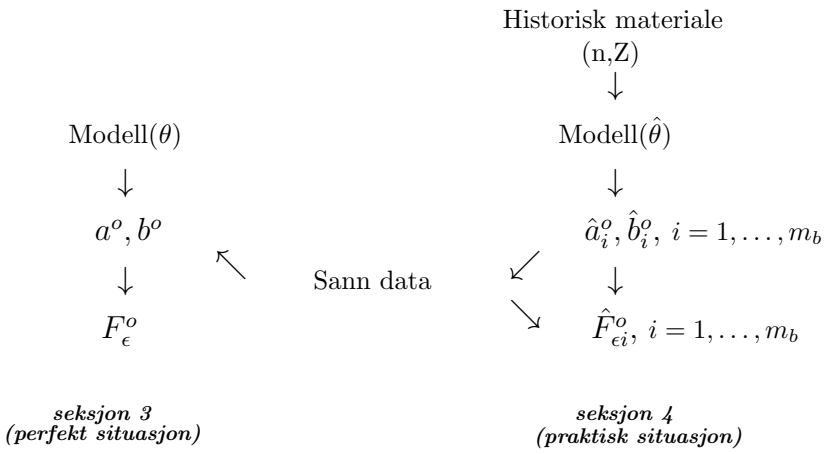
## 4.1 Innledning og tilnærming

I forrige seksjon brukte jeg spesifikke parameterverdier som representerte de sanne parameterverdiene. Jeg hadde i tillegg forutsatt at modellene til den underliggende fordelingen var kjent.

I en praktisk situasjon vet vi hverken hva de sanne parameterverdiene er eller hvilken modell den underliggende fordelingen følger. Vi vil derimot ha et datasatt tilgjengelig som vi kan estimere parametre fra. Jeg vil i denne avhandlingen bruke bootstrap-metoden for å imitere hva som skjer i den virkelige verden og dermed estimere modellparametre på en best mulig måte. Hovedmålet i denne seksjonen er å finne ut kosekvensene av feilene. Vi er spesielt interessert i størrelsen $F_\epsilon(\zeta) = E(G^{ce})/q_\epsilon$. I forrige seksjon fant vi optimerte verdier for reassuranse-grensene $a$ og $b$ som maksimerte denne størrelsen. Disse grensene ble funnet ut ifra den sanne modellen. For å kalkulere estimeringsfeilen kan vi sammenligne forholdstallet fra forrige seksjon med det vi får med bootstrap-metoden i denne seksjonen basert på estimater. Jeg definerer $\zeta$ til å være samlingen av grensene, $\zeta = \{a, b\}$. Jeg setter $\hat{\zeta} = \{\hat{a}, \hat{b}, \}$ til å være grensene som er beregnet fra den sanne modellen. $\hat{\zeta^*} = \{\hat{a^*}, \hat{b^*}\}$ er bootstrap-estimater av grensene basert på historisk data. Den viktige detaljen ved denne prosessen er evalueringen. $\hat{\zeta^*}$ skal evalueres på det sanne datasettet som ga oss de sanne parameterverdiene $\hat{\zeta}$. Det vil føre til at $F_\epsilon(\hat{\zeta^*})$ alltid vil ha en lavere verdi enn $F_\epsilon(\hat{\zeta})$. Noe som er naturlig da $F_\epsilon(\hat{\zeta})$ er den beste verdien under den sanne modellen. Vi skal i denne seksjonen se hvor stor forskjell det er på disse størrelsene. Med andre ord, se hvor stor feilen blir når man estimerer med bootstrap på historiske data kontra estimater fra den sanne modellen, noe som naturligvis ikke er tilgjengelig i praktisk forsikring. Vi vil i dette kapittelet studere tilfellene hvor kravstørrelsene er gamma- og lognormal-fordelte.

## 4.2 Bootstrap

Bootstrapping er en statistisk teknikk som baserer seg på replikasjoner. Vi skal bruke det til estimering av parametrene for modellene vi introduserte i seksjon 3. Det er et enkelt konsept, men siden metoden går ut på å replikere datasettet på estimerte variabler så mange ganger som mulig er bootstrap en robust estimeringsmetode.




Figur 14: *Fremgangsmåte*

I figur 14 har jeg illustrert fremgangsmåten. Til venstre ser vi hvordan vi estimerte parametrene i forrige seksjon. I dette tilfelle tok vi utgangspunktet i sanne parameterverdier. Dette hadde vært en perfekt situasjon. $\theta$ representerer parametrene til poisson, lognormal og gamma. Vi brukte parametrene $\theta = \{\mu, \alpha, \beta\}$ og fikk det sanne datasettet. Vi brukte deretter optimeringsfunksjonen på det sanne datasettet. Dette ga oss de optimerte grensene $a^o$ og $b^o$ som i tur gir oss det optimale forholdstallet $F_\epsilon^o$ under den sanne modellen.

Til høyre i figur 14 ser vi hvordan en praktisk situasjon ser ut. Vi har et historisk datasett ($n$,Z) hvor $n$ er antall observasjoner i det historiske materiale og $Z = z_1, \ldots, z_n$. I et historisk datasett

vil ikke tidsperioden har noe å si for estimatene. Historisk data for 7000 poliser under risiko vil tjene oss like mye som 1000 poliser under risiko over 7 år. Vi estimerer $\hat{\theta}$ fra historisk data. Vi har dermed en estimert modell som vi kan beregne $\hat{a^o}$ og $\hat{b^o}$ påog får $m_b$ bootstrap-replikasjoner av disse grensene. Bootstrap-prosessen er forklart i detalj i algoritme 4.1, Den viktigste delen av metoden er at vi ikke skal evaluere de sistnevnte grensene på den estimerte modellen. Vi skal evaluere $\hat{a^o}$ og $\hat{b^o}$ på det sanne datasettet. Det vil som nevnt tidligere føre til at $\hat{F}_\epsilon^o$ alltid vil være mindre enn $F_\epsilon^o$. Vi vil med dette kunne kaste lys over omfanget av avviket mellom den sanne og estimerte verdien.

Vi er interessert i avviket

$$\pi = F_\epsilon^o - mean\big(\hat{F}_\epsilon^{o*}\big) \tag{4.1}$$

og målet

$$sd(\hat{F}_\epsilon^{o*}) \tag{4.2}$$

der $\hat{F}_\epsilon^{o*}$ er $\hat{F}_{\epsilon 1}^o, \ldots, \hat{F}_{\epsilon m_b}^o$. Størrelsen på det historiske datasettet vil naturligvis påvirke vårt estimat og vi kan intuitivt forstå at $\hat{\theta} \to \theta$ når $n \to \infty$. Dermed vil også

$$\pi = \hat{F}_\epsilon - mean\big(\hat{F}_\epsilon^*\big) \to 0 \,\text{når}\, n \to \infty,\ \text{m} \to \infty \text{ og } m_b \text{ er rimelig stor} \tag{4.3}$$

Deretter er det interessant å se hvor mange observasjoner $n$, simuleringer $m$ og replikasjoner $m_b$ det er behov for, for å oppnå en feil som er innenfor rimelighetens grenser. Antall observasjoner i det historiske datasettet vil naturligvis være avhengig av antall poliser A under risiko og kravintensitet $\mu$ per polise. Dette skal vi se nærmere på i seksjon 4.2. Vi skal bare se på feilen relatert til en begrenset stop-loss-kontrakt hvor kriteriet $F_\epsilon$ (likning 3.7) tar hensyn til vanlig value-at-risk. Studien skal gjøres for gamma- og normal-fordelte totalkrav hvor vi ser på lognormal-fordelte og gamma-fordelte krav størrelser. Det betyr at vi vil ha en sann verdi av $F_\epsilon^o$, a og b tilknyttet lognormal-fordelingen og en for gamma-fordelingen. Det fører til at vi må gjøre en forenkling. Vi antar i det etterfølgende at vi vet den underliggende fordelingen til det historiske materialet. Parametrene er fortsatt ukjent og skal estimeres.

## 4.3 Estimeringsmetode

Vi ser tilbake på figur 14 hvor det første steget ($(n, Z) \to \hat{\theta}$) er parameterestimering fra historisk data. Et vilkårlig historisk materiale består av totalt antall poliser A og observerte krav $z_1, z_2, \ldots, z_n$ hvor alle kravstørrelsen $Z_i$ er uavhengige og identisk fordelte med parameter $\alpha$ og $\beta$. Antall krav $n \sim$ poisson(A$\mu$) med antall poliser A og kravintensitet $\mu$ per polise. I den praktiske verden vil vi, som nevnt tidligere, ikke ha tilgang til de sanne verdiene. Utgangspunktet vil være estimatene basert på historisk materiale. For å estimere $\alpha$ og $\beta$ vil vi bruke momentmetoden som er definert slik

$$\hat{\alpha_1} = E(Z^1), \hat{\alpha_2} = E(Z^2), \ldots, \hat{\alpha_k} = E(Z^k) \qquad \textit{(momentmetoden)} \tag{4.4}$$

hvor $\alpha_i$ er $k$ forskjellige parametre fra samme fordeling. Med utgangspunkt i moment-metoden vil parameterestimatene for lognormal- og gamma-fordelingen være henholdsvis

$$\hat{\alpha} = E(Z), \hat{\beta} = sd(Z) \tag{4.5}$$

og

$$\hat{\beta} = E(Z), \hat{\alpha} = E[(\frac{E(Z)}{sd(Z)})^2] \tag{4.6}$$

Estimatet for kravintensiteten er

$$\hat{\mu} = n/J \tag{4.7}$$

Det neste steget ($\hat{\theta} \rightarrow (\hat{N^*}, \hat{Z^{**}})$) er første bootstrap del hvor vi simulerer $m_b$ replikasjoner av antall krav per portefølje per tidsenhet, $\hat{N^*}$ og $m_b$ sett av kravstørrelser $\hat{Z^*}$ fra de estimerte parametrene, $\hat{\theta} = (\hat{\mu}, \hat{\alpha}, \hat{\beta})$. Se algoritme 4.1. $\hat{Z^*}$ vil ha formatet

$$\hat{Z}^{**} = \hat{Z}_1^*, \ldots, \hat{Z}_{m_b}^*$$

der en replikasjon $i$ er

$$\hat{Z}_i^* = Z_1, \ldots, Z_n \quad i = 1, \ldots, m_b$$

For hvert sett $i$ estimerer vi $\alpha$ og $\beta$ og får $m_b$ bootstrapestimater av $\hat{\alpha}^*$,$\hat{\beta}^*$ og $\hat{\mu}^*$. Det overnevnte steget er representert i linje 5 og 6 i algoritme 4.1.

**Algoritme 4.1** Poisson/normal/gamma-reassuranse bootstrap

```
1   Inn: m, m_b, α̂, β̂, μ̂, J
2
3   #Monte Carlo med boostrap-estimater
4   for i = 1, 2, ..., m_b
5       N* ~ poisson(Jμ̂),  Z* ~ normal/gamma(α̂, β̂)
6       μ̂^{*bs} ← N*/J,  [α̂^{*bs}, β̂^{*bs}] ← Z*
7       for j = 1, 2, ..., m
8           N = poisson(Jμ̂_i^{bs})
9           Z = normal/gamma(N,α̂_i^{bs},β̂_i^{bs})
10          X_ij = sum(Z)
11      [â_i, b̂_i] = optim(F_ε(X_i.))
12  return  â*, b̂*
```

Vi kan se fra algoritme 4.1 at den innerste for-løkken er identisk med for-løkken i porteføljekonstruksjonen i kap.2.2. Forskjellen er at vi i dette tilfelle bruker bootstrap-estimater og ikke sanne verdier. Den ytterste for-løkken sørger for å lage $m_b$ replikasjoner av et datasett av agreggerte krav X med $m$ simuleringer. De $m$ simuleringene blir laget i den innerste løkken.

$$\hat{X}_{ij}^{**} = X_{1j}^*, X_{2j}^*, \ldots, X_{m_b j}^*, \quad j = 1, \ldots, m$$

hvor en vilkårlig $i$ er $m$ samlinger av aggregerte krav

$$X_{ij}^* = X_{i1}, X_{i2}, \ldots, X_{im}, \quad i = 1, \ldots, m_b$$

der hver X er summen av $N_j$ krav

$$X_{ij} = \sum_{k=1}^{N_j} Z_k, \quad k = 1, \ldots, N_j$$

Dette betyr at vi får $m * m_b$ simuleringer av aggregerte krav X. Til slutt skal vi finne verdier av a og b for hver $i$ som som optimererer kriterie $F_\epsilon$ på tilsvarende datasett $X_{i.}^*$ ved hjelp av funksjonen optim. Det ser vi på linje 11 i algoritmen. Her betyr **.** i indekseringen at j er indeksert over 1:m. Algoritmen returnerer $m_b$ bootstrap-estimater av kontraktsparameterene $\hat{a}^*$, $\hat{b}^*$. Disse vil bli evaluert på det sanne datasettet og vi kan dermed beregne likning 4.1 og 4.2.

## 4.4 Estimeringsfeil

Med all verktøy på plass for å beregne optimal reassuranse vil vi ta en nærmere titt på hvordan feilen (likning 4.1) og standardavviket (likning 4.2) forandrer seg i takt med endringer i forskjellige variabler. Antall poliser A i historisk data, antall observasjoner $n$ og antall bootstrap-replikasjoner $m_b$ vil varieres. Før jeg fremstiller resultatene vil jeg gi skrive litt om Monte Carlo-feil som forekommer i algoritmen over. Vårt mål er å kvantifisere reassuransemålet $F_\epsilon^o = F_\epsilon(\theta)$. Som tidligere nevnt må vi nøye oss med et estimat, $\hat{F}_\epsilon^o = F_\epsilon(\hat{\theta})$. I en Monte Carlo-prosess må vi estimere $\hat{\theta}$ fra historiske data og beregne bootstrap-estimatet $\hat{\theta}^*$ som igjen brukes til å kvantifisere kriterie $\hat{F}_\epsilon^{o*} = F_\epsilon(\hat{\theta}^*)$. Vi vil med dette få tre feil.

- $F_\epsilon^o - \hat{F}_\epsilon^{o*}$ total feil
- $\hat{F}_\epsilon^o - \hat{F}_\epsilon^{o*}$ Monte Carlo-feil ($m_b$ replikasjoner og $m$ simuleringer)
- $F_\epsilon^o - \hat{F}_\epsilon^o$ estimeringsfeil ($n$ observasjoner)

Feilen på linje to kommer kun fra Monte Carlo-delen. Det viser avviket mellom det estimerte målet og bootstrap-estimatene. Linje tre er feilen som oppstår ved estimering av parametrene fra historiske data, avviket mellom $\hat{\theta}$ og $\theta$. Den totale feilen er avviket mellom bootstrap-estimatet og den sanne verdien. Som nevnt tidligere blir det totalt $m * m_b$ simuleringer av aggregerte krav inkludert optimering av funksjonen $F_\epsilon$, $m_b$ ganger. Dette krever mye maskinkraft. Det må vurderes hvor mange simuleringer og replikasjoner det skal brukes. Tallet $n$ har vi ikke noe kontroll over i praksis og vi må derfor nøye oss med det antalle observasjoner vi har fra et historisk datasett. For feilanalysens skyld skal vi variere antall historiske observasjoner for å se estimeringsfeilens påvirkning på reassuransemålenet. Antall simuleringer $m$, antall aggregerte krav simulert per boostrap-estimat, må maksimeres så mye det lar seg gjøre med tanke på maskinens kraft. For antall boostrap-replikasjoner $m_b$ er det ikke nødvendig med spesielt høye verdier i følge spesialisert boostrap-litteratur. Det er gjentakende eksempler på boostrap-simuleringer hvor det brukes nærmere 1000 og 10000 bootstrap-replikasjoner; Bølviken (2014). Det nevnes også nærmere i samme litteratur at det holder med rundt $m_b$=50 der det er referert videre til bootstrap-litteraur; Efron og Tibshirani(1993).

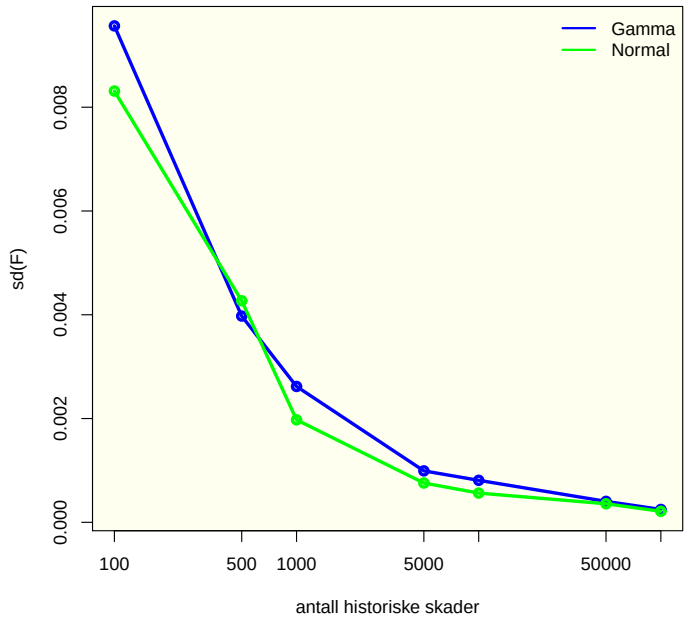
## 4.5 Resultater

For å gjøre en kvalitetssjekk programmerte min veileder og jeg uavhenging på forskjellige plattformer. Vi brukte de samme parametrene og sikret oss at vi fikk de samme resultatene. Samtidig var det nødvendig for meg å ha koden min tilgjengelig på en annen platform enn R. I og med at beregningene og simuleringene krevde mye maskinkraft så var R for langsom. For relevante parameterverdier og fornuftig antall simuleringer kunne det ta timer før programmet var kjørt ferdig. Erik Bølviken programmerte i Fortran, et programmeringsspråk som gjør beregningene i en mye høyere hastighet. Dette var nødvendig for at jeg effektivt skulle kunne analyse estimeringsfeilene. I resultattabellene skal jeg føre opp tiden hver simulering tok. Vi vil se at den lengste simuleringstiden på Fortran med $m_b$=1000, og m=100000 var på 47 minutter og 20 sekunder. Tilsvarende simuleringer i R tok over 9 timer.
I tabellen er feilene listet opp i prosent både for tilfellet hvor kravstørrelsene er lognormal- og gamma-fordelt.

| $m_b$=50 | m=100000 | Gamma $\rightarrow \alpha = 1, \beta = 3$ | | Lognormal $\rightarrow \alpha = 1, \beta = 0.5$ |
|---|---|---|---|---|
| | Gamma | | Lognormal | |
| **Historisk data** | $\mathrm{E}[\hat{F}_\epsilon^{o*}]$ | $\pi$ | $\mathrm{E}[\hat{F}_\epsilon^{o*}]$ | $\pi$ |
| n = 100 | 0.0780 | 0.0109 | 0.0844 | 0.0067 |
| n = 500 | 0.0853 | 0.0036 | 0.0876 | 0.0035 |
| n = 1000 | 0.0866 | 0.0023 | 0.0892 | 0.0018 |
| n = 5000 | 0.0880 | 0.0009 | 0.0905 | 0.0005 |
| n = 10000 | 0.0882 | 0.0007 | 0.0907 | 0.0003 |
| n = 50000 | 0.0886 | 0.0003 | 0.0909 | 0.0002 |
| n = 100000 | 0.0887 | 0.0002 | 0.0909 | 0.0001 |
| **Sann verdi** | **0.088891** | | **0.091066** | |

Tabell 4.1: *Avvik og estimerte kriterieverdier gitte verdier av n*

I denne simuleringen satte vi et fast antall simuleringer m=100000. Vi ser fra tabellen at samtlige $\pi$-verdier reduseres jo større historisk materiale vi har. Det ser ut til at den stabiliserer seg rundt et sted over 0. Det er fordi avviket $\pi$ ser på den totale feilen nevnt tidligere i avsnittet. Det er fortsatt en underliggende Monte Carlo-feil som ikke forsvinner helt. I figur 15 ser vi på standardavviket til bootstrap-estimatet av kriterie $\hat{F}_\epsilon^*$. Vi ser en reduksjon i standardavviket for større verdier av $n$. Det er en sterk effekt av størrelsen på det historiske materialet både på avviket $\pi$ og $\mathrm{sd}(F_\epsilon)$. Som nevnt i seksjon 4.4 er estimeringsfeilen den største kilden til det totale avviket. Antall observasjoner $n$ vil ha en sterk effekt på $\pi$.




Figur 15: *$sd(\hat{F}_\epsilon^o)$ for gitte verdier av n*

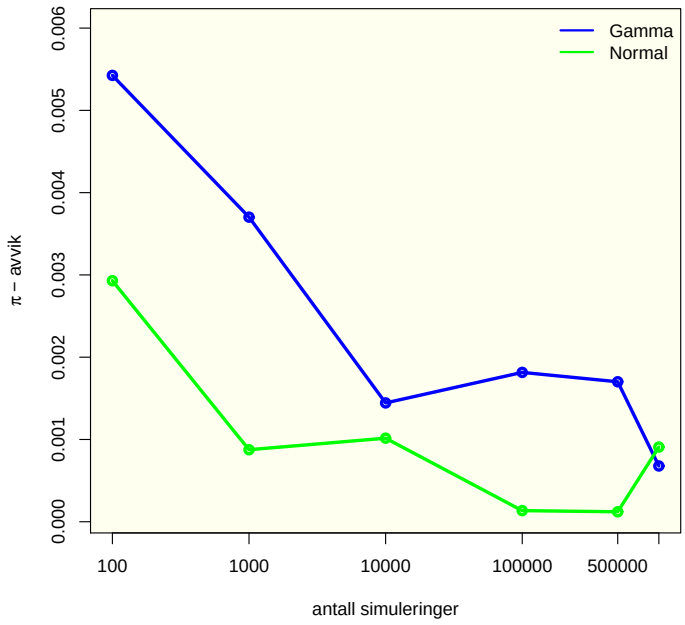
I en perfekt verden vil $\pi$ bli nærmest 0 når vi øker $m$ og $n$. Hvor nær 0 kommer feilen hvis vi øker antall simulering $m$ itillegg til å ha et høyt antall historisk materiale tilgjengelig? Det som er interessant å tenke på før vi varierer $m$ er at, uansett hvor høyt antall simuleringer vi har så vil ikke feilen gå mot 0 så lenge vi ikke har et stort nok historisk materiale. Har vi for eksempel et lavt antall historiske krav og vi har veldig mange simuleringer $m$ så resulterer det i

at bootstrap-estimatene blir konsistente rundt estimatene som i utgangspunktet er feil. En stor $m$ vil med andre ord redusere $\hat{\theta}$-mean($\hat{\theta^*}$) men feilen $\pi$ vil ikke nødvendigvis bli noe mindre som en direkte konsekvens av kun en økning i $m$.

| $m_b$=50 | n=100000 | Gamma $\rightarrow \alpha = 1, \beta = 3$ | | Lognormal $\rightarrow \alpha = 1, \beta = 0.5$ |
|---|---|---|---|---|
| | Gamma | | Lognormal | |
| **Simuleringer** | $\mathrm{E}[\hat{F}_\epsilon^{o*}]$ | $\pi$ | $\mathrm{E}[\hat{F}_\epsilon^{o*}]$ | $\pi$ |
| m=100 | 0.0835 | 0.0054 | 0.0881 | 0.0029 |
| m=1000 | 0.0852 | 0.0037 | 0.0902 | 0.0009 |
| m=10000 | 0.0875 | 0.0014 | 0.0901 | 0.0010 |
| m=100000 | 0.0871 | 0.0018 | 0.0909 | 0.0001 |
| m=500000 | 0.0872 | 0.0017 | 0.0910 | 0.0001 |
| m=1000000 | 0.0882 | 0.0007 | 0.0902 | 0.0009 |
| **Sann verdi** | **0.088891** | | **0.091066** | |

Tabell 4.2: *Avvik og estimerte kriterie verdier for gitte verdier av m*

Vi ser fra tabell 4.2 at avviket $\pi$ reduseres jo flere simuleringer $m$ vi bruker. Avviket reduseres ikke like kraftig som med en økning i historisk materiale men det er fortsatt nevneverdig forskjeller. Vi ser samtidig at avviket for poisson-/lognormal-modellen er noe mindre enn poisson-/gamma-modellen. Det ser vi tydelig i figur 16 hvor den grønne kurven ligger under den blå for alle verdier av $m$. Ut ifra tabellen og figuren ser det ut som at 100 000 er et fornuftig minimum for antall simuleringer. I tabell 4.3 kan vi se resultater med forskjellige antall boostrap-replikasjoner $m_b$. For gamma-modellen ser vi en kraftig reduksjon i sd($F_\epsilon^{o*}$) jo flere antall replikasjoner vi har.




Figur 16: $\pi$ *for gitte verdier av m*

Dette er Monte Carlo-feilen nevnt i seksjon 4.4. For lognormal-modellen ser vi en mer beskjeden reduksjon for standardavviket. Avviket $\pi$ forandrer seg ikke mye for høyere antall replikasjoner. Det spørs dermed om det lønner seg å ha et høyt antall replikasjoner $m_b$. For lognormal-modellen

ser det ikke ut til at det gir stort utslag på avviket. For gamma-modellen får vi en reduksjon for både $\pi$ og $\text{sd}(F_\epsilon^{o*})$. For å bestemme antall replikasjoner må man vurdere nytten opp mot simuleringstiden. Hvis grunnsimuleringene er krevende i utgangspunktet kan en stor $m_b$ føre til upassende lang simuleringstid.

| **m=100000** | **n=100000** | **Gamma** $\rightarrow \alpha = 1, \beta = 3$ | | **Lognormal** $\rightarrow \alpha = 1, \beta = 0.5$ | | |
|---|---|---|---|---|---|---|
| **Bootstrap** | **Gamma** | | | **Lognormal** | | |
| **replikasjoner** | $\text{E}[\hat{F}_\epsilon^{o*}]$ | $\pi$ | $\text{sd}(\hat{F}_\epsilon^{o*})$ | $\text{E}[\hat{F}_\epsilon^{o*}]$ | $\pi$ | $\text{sd}(\hat{F}_\epsilon^{o*})$ |
| $m_b$=50 | 0.08713 | 0.00176 | 0.00776 | 0.09011 | 0.00096 | 0.00447 |
| $m_b$=100 | 0.08789 | 0.00100 | 0.00552 | 0.090071 | 0.000995 | 0.00430 |
| $m_b$=1000 | 0.08793 | 0.00096 | 0.00510 | 0.090076 | 0.000999 | 0.00426 |
| **Sann verdi** | **0.0888796** | | | **0.0910713** | | |

Tabell 4.3: *Avvik og estimerte kriterieverdier for gitte verdier av $m_b$*

Videre er jeg interessert i å se på effekten av forskjellig parametre på avviket og standardavviket. Parametrene $\alpha$ og $\beta$ påvirker størrelsen på hvert krav Z. Som nevnt i starten av avhandlingen så er $\alpha$=shape og $\beta$=scale for gamma-modellen mens for den lognormale-modellen står $\alpha$ for forventningen og $\beta$ for standardavviket. For de 3 første radene i tabell 4.4 ser jeg på økningen av $\beta$ for begge modellene.

| | | **m=100000** | | **n=100000** | $m_b$**=50** | | |
|---|---|---|---|---|---|---|---|
| **Gamma** | | | | **Log-normal** | | | |
| $(\alpha,\beta)$ | $\text{E}[\hat{F}_\epsilon^{o*}]$ | $F_\epsilon^o$ | $\pi$ | $(\alpha,\beta)$ | $\text{E}[\hat{F}_\epsilon^{o*}]$ | $F_\epsilon^o$ | $\pi$ |
| $(1,2)$ | 0.08713 | 0.08828 | 0.00116 | (1,0.5) | 0.09031 | 0.09113 | 0.00082 |
| $(1,3)$ | 0.08713 | 0.08828 | 0.00116 | (1,0.7) | 0.08906 | 0.09003 | 0.00097 |
| $(1,4)$ | 0.08713 | 0.08828 | 0.00116 | (1,1.0) | 0.08619 | 0.08730 | 0.00111 |
| $(2,2)$ | 0.08970 | 0.09041 | 0.00072 | (2,0.5) | 0.09031 | 0.09113 | 0.00082 |
| $(3,2)$ | 0.09008 | 0.09070 | 0.00062 | (3,0.5) | 0.09031 | 0.09113 | 0.00082 |
| $(4,2)$ | 0.09015 | 0.09126 | 0.00111 | (4,0.5) | 0.09031 | 0.09113 | 0.00082 |

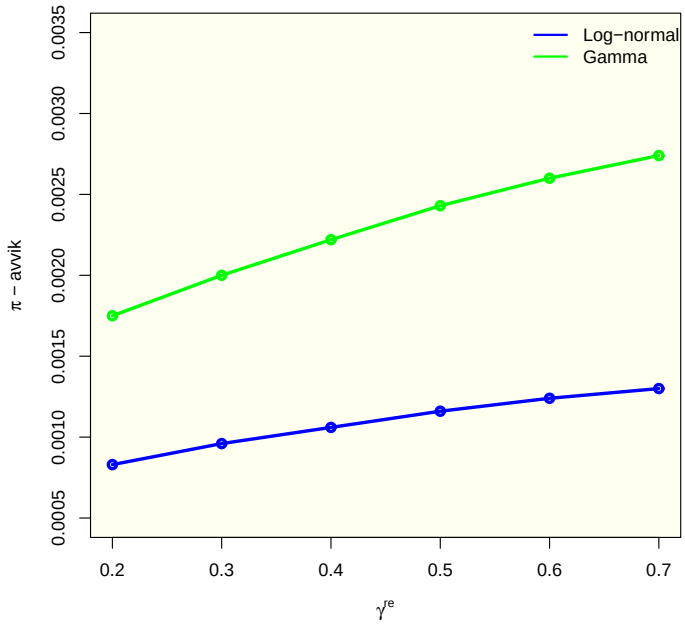
Tabell 4.4: *Avvik og estimerte kriterieverdier for gitte parameter verdier*

Det første man legger merke til er at resultatene for gamma-modellene for de tre første linjene er identiske. Det kommer av at scale-parameteren ikke har noe effekt på forholdstallet $F_\epsilon^o$. En oppskalering av kravstørrelsene resulterer i at de optimale grensene $a$ og $b$ også skaleres opp. Samtidig som vi vet at forutsetningen a+b=$q_\epsilon$ alltid skal være oppfylt så betyr det at en oppskalering av kravstørrelsen ikke endrer forholdet mellom cedent og reassurandør. Dermed får vi ingen endring i forholdstallet. Dette betyr dermed at avviket $\pi$ heller ikke blir påvirket. Ser vi på de tre første linjene for lognormal-modellen ser vi en reduksjon i forsholdstallet og økning i avviket $\pi$. Det er fordi $\beta$ for lognormal-modellen tilsvarer standardavviket. Endringer i den parameteren vil endre forholde mellom cedent og reassurandør da risikoen endrer seg. På de siste tre linjene er rollene reversert. Nå ser vi på tilfellet der vi øker $\alpha$. $\alpha$ står for forventningen til lognormal-modellen og shape for gamma-modellen. Vi ser dermed at resultatene for lognormal-modellen er identiske for forskjellige verdier av $\alpha$ men forskjellig for gamma-modellen.

| **m=100000** | **n=100000** | | **Gamma** $\rightarrow \alpha = 1, \beta = 3$ | | **Lognormal** $\rightarrow \alpha = 1, \beta = 0.5$ |
|---|---|---|---|---|---|
| | **Gamma** | | | **Lognormal** | |
| | $\mathrm{E}[\hat{F}_\epsilon^{o*}]$ | $F_\epsilon^o$ | $\pi$ | $\mathrm{E}[\hat{F}_\epsilon^{o*}]$ | $F_\epsilon^o$ | $\pi$ |
| $\gamma^{re}$=0.2 | 0.08713 | 0.08888 | 0.00175 | 0.09031 | 0.09114 | 0.00083 |
| $\gamma^{re}$=0.3 | 0.08347 | 0.08547 | 0.00200 | 0.08738 | 0.08834 | 0.00096 |
| $\gamma^{re}$=0.4 | 0.08130 | 0.08352 | 0.00222 | 0.08565 | 0.08671 | 0.00106 |
| $\gamma^{re}$=0.5 | 0.07977 | 0.08220 | 0.00243 | 0.08444 | 0.08560 | 0.00116 |
| $\gamma^{re}$=0.6 | 0.07861 | 0.08121 | 0.00260 | 0.08353 | 0.08477 | 0.00124 |
| $\gamma^{re}$=0.7 | 0.07769 | 0.08043 | 0.00274 | 0.08281 | 0.08411 | 0.00130 |

Tabell 4.5: *Avvik og estimerte kriterieverdier for gitte reassuranse priser*

I forrige kapittel så vi på hvordan de optimale grensene endret seg for forskjellige priser for reassurandør, $\gamma^{re}$. Ladningen er som nevnt tidligere prispåslaget cedenten må betale for å overføre risiko til reassurandør. Vi så i tabell 3.5 i kapittel 3 at cedenten overførte mindre ansvar over til reassurandør i takt med et økende prispåslag fra reassurandør. Resultatene i tabell 4.5 viser en konstant økning i avviket $\pi$. Jo høyere prispåslaget er jo høyere blir avviket samtidig som forholdstallet reduseres, noe som vi også så i forrige kapittel.




Figur 17: *Avvik for gitte reassuranse-priser*

# 5 Konklusjon

Vi begynte med å introdusere to type reassuranse-kontrakter som skulle analyseres i avhandlingen. Henholdsvis enkel stop-loss og begrenset-stop loss. De ble vist at disse kontraktene optimerer forventet gevinst mot nedsiderisiko hvor vanlig value-at-risk og betinget value-at-risk ble betraktet. Som det ble nevnt i innledningen så er det ikke vanlig praksis idag å numerisk optimere reassuranse-kontrakter med hensyn på kriterier som selskaper kan være interessert i. Det testes for forskjellige alternativer med og uten reassuranse før man velger alternativet som gir mest lønnsomhet for en portefølje. I praksis vil det samtidig være flere faktorer enn bare ett kriterie

å ta hensyn til. Selskaper må ta hensyn til hvilken strategier og mål selskapet har satt. Skal de vokse eller bli mer lønnsomme. I denne avhandlingen tok vi ikke hensyn til slike faktorer. Vi forenklet problemstillingen ned til optimering av kun ett kriterie.

Etter å ha gitt en introduksjon til reassuranse-kontraktene og grunnleggende skadeforsikring var det interessant å belyse effekten av den underliggende modellen på reassuranse-kontraktene. Siden reassuranse-kontraktene brukes på datasett av aggregerte krav var det interessant å se på tetthetssansynlighetene til de. Vi såat jo større spredning det var i de aggregerte kravstørrelsene jo større ble parameterverdien $b$. Parameterverdien $a$ økte i takt med forventningen til datasettet. Parameterverdien $b$ er lengden på intervallet fra $a$ til $a+b$ hvor cedenten står til ansvar for en fast sum $a$ og reassurandør tar over øvrig risiko. Kravstørrelser frem til $a$ var cedentens ansvar. Med andre ord vil cedenten stå til ansvar for hele kravet hvis størrelsen er under $a$, deretter en fast sum $a$ hvis den aggregerte kravstørrelsen overstiger $a$ og er mindre enn $a + b$. Ved hjelp av Markovitz-teorien viste vi at de optimale kontraktgrensene skal tilfredstille $a + b{=}q_\epsilon$. Dette betyr at reassurandørs maksimale ansvar ikke vil overskride den fastsatte (1-$\epsilon$)%-reserven. Slik vil kontraktparametrene $a$ og $b$ endre seg i takt med den underliggende sannsynlighetstettheten for at kriteriet skal holde seg på Markovitz-grensen, hvor selskapet oppnår den beste kombinasjonen av forventet gevinst og nedsiderisiko. Vi fant deretter ut effekten av prisen for reassurandør førte til et mindre forholdstall samtidig som cedenten overførte mindre risiko i takt med en økende pris.

Det var nærliggende å sammenlikne disse optimale reassuranse-kontraktene mot en proporsjonal reassuranse-kontrakt. En optimal kriterieverdi for en proporsjonal-kontrakt er tilsvarende til en situasjon hvor man ikke har reassuranse. I tabell 3.5 viste vi at kriteriet med en begrenset stop-loss-kontrakt var 0.0147 bedre enn en proporsjonal-kontrakt i tilfellet hvor kravstørrelsene var gamma-fordelte. Kriteriet var også bedre for alle de andre type porteføljene vi analyserte. Resultatene viste også at kapitalreserven som må avsettes var betraktelig mindre med en optimal reassuranse-kontrakt. Med dette viste vi at et selskap kan frigjøre kapital til videre forvaltning hvis man bruker en begrenset stop-loss-kontrakt. Men det er også klart at inntjeningspotensiale reduseres i takt med økende andel av reassuranse da reassurandør skal ha en del av premien for risikoen de påtar seg.

Til slutt så vi på estimeringsfeil ved disse beregningene. I og med at vi tok i bruk Monte Carlo i bootstrap-estimatene var det essensielt å analysere hvordan estimeringsfeilen forandret seg i takt med historisk materiale $n$, antall simuleringer $m$, og antall bootstrap-replikasjoner $m_b$. En gjentakende trend i kapittel 4 var at estimeringsfeilene var større for gamma-modellen. For lognormal sin del så reduserte avviket fortere etterhvert som man økte $n$ og $m$. Vi så også at størrelsen på det historiske materialet, $n$, var den største kilden til avvik i estimatene. En stor $n$ reduserte både avviket, likning 4.1 og standardavviket, likning 4.2, betydelig. Antall simuleringer $m$ bør holdes minimum 100 000 ettersom det vil påvirke avviket. Antall bootstrap-replikasjoner, $m_b$, blir en vurderingssak mellom nytten og simuleringstiden. Antall bootstrap-replikasjoner $m_b$ viste seg å ha en betydelig effekt på estimatene for gamma-modellen. For lognormal-modellen var det ikke store endringer for forskjellige verdier av $m_b$.

# 6 Vedlegg

```
----------
Kapittel 2
----------
```

**Funksjon som plotter en kontrakts figur for enkel og begrenset stop loss gitt et datasett og parametere**

---

```
stop_loss = function(X_agg,type,kon,d,lim){


if(missing(kon)){kon=c(90,20)}
if(missing(d)){d=90}
if(missing(lim)){lim=c(80,120)}

m = length(X_agg)

a=kon[1]
b=kon[2]

lim1 = lim[1]
lim2 = lim[2]

X_ce = rep(0,m)
X_re = rep(0,m)

if(type=="lim"){

for(i in 1:m){

if(X_agg[i]<=a){X_ce[i]=X_agg[i]}
else if(X_agg[i]<=a+b & X_agg[i]>a){X_ce[i]=a;X_re[i]=X_agg[i]-a}
else if(X_agg[i]>a+b){X_re[i]=b;X_ce[i]=X_agg[i]-b}
else{print("Feeeil")}

}
par(mfrow=c(1,2))
plot(sort(X_agg),sort(X_re),xlab="krav",ylab="reassuranse",main="AxB kontrakt",
type="l",col="blue",xlim=c(0,max(X_agg)+20),ylim=c(0,max(X_ce)))
legend("topleft",c("A","A+B"),col=c("green","red"),lty=1,cex=0.7)
plot(sort(X_agg),sort(X_ce),xlab="krav",ylab="cedent",type="l",col="blue")
abline(v=a,col="green")
abline(v=a+b,col="red")


}
```

```
if(type=="trunc"){

for(i in 1:m){

if(X_agg[i]<lim1){X_re[i]=0;X_ce[i]=X_agg[i]}
if(X_agg[i]<=lim2 & X_agg[i]>lim1){
X_re[i]=max(X_agg[i]-lim1,0);X_ce[i]=X_agg[i]-max(X_agg[i]-lim1,0)
}
if(X_agg[i]>lim2){X_re[i]=0;X_ce[i]=X_agg[i]}

}
plot_re=X_re[X_re!=0]

par(mfrow=c(1,2))
plot(seq(lim1,lim2,length.out=length(plot_re)),sort(plot_re),xlab="krav",
ylab="reassuranse",main="Avkortet stop loss",type="l",col="blue",
xlim=c(0,max(X_agg)+20),ylim=c(0,max(X_ce)))
legend("topleft",c("Reassuransen vre grense","Reassuransens minste grense"),
            col=c("red","green"), lty=1,cex=0.7)

plot(sort(X_agg),sort(X_ce),xlab="krav",ylab="cedent",type="l",col="blue")
abline(v=lim1,col="green")
abline(v=lim2,col="red")


}

if(type=="stop"){

for(i in 1:m){
X_re[i]=max(X_agg[i]-d,0)
X_ce[i]=X_agg[i]-max(X_agg[i]-d,0)
}
par(mfrow=c(1,2))
plot(sort(X_agg),sort(X_re),xlab="krav",ylab="reassuranse",main="Borch sin stop loss",
type="l",col="blue",ylim=c(0,max(X_agg)),xlim=c(0,max(X_agg)+20))
legend("topleft",c("Reassuransens nedre grense"),col="green",lty=1,cex=0.7)
plot(sort(X_agg),sort(X_ce),xlab="krav",ylab="cedent",type="l",
col="blue",ylim=c(min(X_ce),max(X_agg)),
xlim=c(0,max(X_agg)))
abline(v=d,col="green")




}

ro_re = 1.4
ro_ce = 1.2

pi_ce=mean(X_ce)
```

```
pi_re=mean(X_re)

EG = (pi_ce*ro_ce)-(pi_re*ro_re)

res=sort(X_agg)[0.99*m]

res_ce=sort(X_ce)[0.99*m]

cat("-----------------------------------------|","\n")
cat("Average claim size per year -->", mean(X_agg),"\n")
cat("-----------------------------------------|","\n")
cat("Reinsurance premium -->", pi_re,"\n")
cat("-----------------------------------------|","\n")
cat("Cedent premium -->", pi_ce,"\n")
cat("-----------------------------------------|","\n")
cat("Average Reinsurance Percentage -->", (mean(X_re)/mean(X_agg))*100,"\n")
cat("-----------------------------------------|","\n")
cat("99 prosent reserve -->",res ,"\n")
cat("-----------------------------------------|","\n")
cat("cedents 99 prosent reserve -->",res_ce ,"\n")
cat("-----------------------------------------|","\n")

return(list(res_ce,EG,pi_ce,X_agg,X_ce,X_re))


}
```

```
----------
Kapittel 3
----------
```

## Simulerer gaussiske porteføljer

```
m=100000

N1=normal_data(m=m,mu=0.5,theta=0.8)[[1]]
N2=normal_data(m=m,mu=0.5,theta=0.5)[[1]]
N3=normal_data(m=m,mu=0.5,theta=0.2)[[1]]
N4=normal_data(m=m,mu=0.8,theta=0.5)[[1]]
N5=normal_data(m=m,mu=0.5,theta=0.5)[[1]]
N6=normal_data(m=m,mu=0.2,theta=0.5)[[1]]
N7=normal_data(m=m,mu=0.8,theta=0.8)[[1]]
N8=normal_data(m=m,mu=0.5,theta=0.5)[[1]]
N9=normal_data(m=m,mu=0.2,theta=0.2)[[1]]
```

## Tetthetsplot for hver portefølje

```
zettl1 = as.expression(bquote(mu~" = 0.5,"~sigma~"= 0.8"))
zettl2 = as.expression(bquote(mu~" = 0.5,"~sigma~"= 0.5"))
zettl3 = as.expression(bquote(mu~" = 0.5,"~sigma~"= 0.2"))
zettl4 = as.expression(bquote(mu~" = 0.8,"~sigma~"= 0.5"))
zettl5 = as.expression(bquote(mu~" = 0.5,"~sigma~"= 0.5"))
zettl6 = as.expression(bquote(mu~" = 0.2,"~sigma~"= 0.5"))
zettl7 = as.expression(bquote(mu~" = 0.8,"~sigma~"= 0.8"))
zettl8 = as.expression(bquote(mu~" = 0.5,"~sigma~"= 0.5"))
zettl9 = as.expression(bquote(mu~" = 0.2,"~sigma~"= 0.2"))

par(mfrow=c(1,3))
plot(density(N1),col="red",lwd=2,main="",xlab="Aggregerte krav",xlim=c(80,550),
ylim=c(0,0.032))
rect(par("usr")[1],par("usr")[3],par("usr")[2],par("usr")[4],col = "ivory")
points(density(N1),type="l",col="red",lwd=2)
lines(density(N2),col="blue",lwd=2)
lines(density(N3),col="green",lwd=2)
legend("topright",c(zettl1,zettl2,zettl3),fill=c("red","blue","green"),cex=1,bty = "n")

plot(density(N4),col="red",lwd=2,main="",xlab="Aggregerte krav",xlim=c(80,550),
ylim=c(0,0.032))
rect(par("usr")[1],par("usr")[3],par("usr")[2],par("usr")[4],col = "ivory")
points(density(N4),type="l",col="red",lwd=2)
lines(density(N5),col="blue",lwd=2)
lines(density(N6),col="green",lwd=2)
legend("topright",c(zettl4,zettl5,zettl6),fill=c("red","blue","green"),cex=1,bty = "n")

plot(density(N7),col="red",lwd=2,main="",xlab="Aggregerte krav",xlim=c(80,550),
ylim=c(0,0.032))
rect(par("usr")[1],par("usr")[3],par("usr")[2],par("usr")[4],col = "ivory")
points(density(N7),type="l",col="red",lwd=2)
lines(density(N8),col="blue",lwd=2)
lines(density(N9),col="green",lwd=2)
legend("topright",c(zettl7,zettl8,zettl9),fill=c("red","blue","green"),cex=1,bty = "n")
```

### Beregner optimal kontrakt

```
ndata=N6

per1 = (max(ndata)-min(ndata))/3

#Optimerer begrenset stop loss
opt_par = optim(c(mean(ndata),per1),pi_div_q_lim,X_agg=ndata,control=list(fnscale=-1))

#Optimerer Enkel Stop Loss
opt_par_d = optimize(pi_div_q_enkel,interval=c(min(ndata),max(ndata)),
X_agg=ndata,maximum=TRUE)
```

### Utskrift av estimatene

```
cat("99 og min,",sort(ndata)[length(ndata)*0.99],"  ",min(ndata),"\n")
cat("Mean og SD,",mean(ndata),"  ",sd(ndata),"\n")
cat("Begrenset Stop Loss,","A =",opt_par$par[1],"B =",opt_par$par[2],"\n")

opt_par$par[1]+opt_par$par[2]
sort(ndata)[length(ndata)*0.99]

cat("Enkel Stop Loss, D =",opt_par_d$maximum,"\n")
##################################################
```

### Simulerer gamma porteføljer

```
gl1 = as.expression(bquote(alpha~"= 1,"~beta~"= 1"))
gl2 = as.expression(bquote(alpha~"= 1,"~beta~"= 2"))
gl3 = as.expression(bquote(alpha~"= 1,"~beta~"= 3"))
gl4 = as.expression(bquote(alpha~"= 1,"~beta~"= 4"))
gl5 = as.expression(bquote(alpha~"= 3,"~beta~"= 3"))
gl6 = as.expression(bquote(alpha~"= 2,"~beta~"= 2"))
gl7 = as.expression(bquote(alpha~"= 2,"~beta~"= 1"))
gl8 = as.expression(bquote(alpha~"= 3,"~beta~"= 1"))
gl9 = as.expression(bquote(alpha~"= 4,"~beta~"= 1"))

m=100000

#Simulerer
g1=gamma_data(m=m,alpha=1,beta=1)[[1]]
g2=gamma_data(m=m,alpha=1,beta=2)[[1]]
g3=gamma_data(m=m,alpha=1,beta=3)[[1]]
g4=gamma_data(m=m,alpha=1,beta=4)[[1]]
g5=gamma_data(m=m,alpha=3,beta=3)[[1]]
g6=gamma_data(m=m,alpha=2,beta=2)[[1]]
g7=gamma_data(m=m,alpha=2,beta=1)[[1]]
g8=gamma_data(m=m,alpha=3,beta=1)[[1]]
g9=gamma_data(m=m,alpha=4,beta=1)[[1]]
```

### Tetthetsplot for hver portefølje

```
par(mfrow=c(1,3))

plot(density(g2),col="red",main="",xlim=c(0,700),xlab="Aggregerte krav",
ylim=c(0,0.027))
rect(par("usr")[1],par("usr")[3],par("usr")[2],par("usr")[4],col = "ivory")
points(density(g2),type="l",col="red",lwd=2)
lines(density(g3),col="blue",lwd=2)
lines(density(g4),col="green",lwd=2)
```

```
legend("topright",c(gl2,gl3,gl4),fill=c("red","blue","green"),cex=1,bty = "n")
abline(h=0)

plot(density(g7),col="red",main="",xlim=c(0,700),xlab="Aggregerte krav",
ylim=c(0,0.027))
rect(par("usr")[1],par("usr")[3],par("usr")[2],par("usr")[4],col = "ivory")
points(density(g7),type="l",col="red",lwd=2)
lines(density(g8),col="blue",lwd=2)
lines(density(g9),col="green",lwd=2)
legend("topright",c(gl7,gl8,gl9),fill=c("red","blue","green"),cex=1,bty = "n")
abline(h=0)

plot(density(g1),col="red",main="",xlim=c(0,1200),xlab="Aggregerte krav",
ylim=c(0,0.029))
rect(par("usr")[1],par("usr")[3],par("usr")[2],par("usr")[4],col = "ivory")
points(density(g1),type="l",col="red",lwd=2)
lines(density(g6),col="blue",lwd=2)
lines(density(g5),col="green",lwd=2)
legend("topright",c(gl1,gl6,gl5),fill=c("red","blue","green"),cex=1,bty = "n")
abline(h=0)
```

### Beregner optimal kontrakt

```
ndata=g9

#Startverdier
per1 = (max(ndata)-min(ndata))/3

#Optimerer begrenset stop loss
opt_par = optim(c(mean(ndata),per1),pi_div_q_lim,X_agg=ndata,control=list(fnscale=-1))

#Optimerer Enkel Stop Loss
opt_par_d = optimize(pi_div_q_enkel,interval=c(min(ndata),max(ndata)),
X_agg=ndata,maximum=TRUE)
```

### Utskrift av parameterene

```
cat("99 og min,",sort(ndata)[length(ndata)*0.99],"  ",min(ndata),"\n")
cat("Mean og SD,",mean(ndata),"  ",sd(ndata),"\n")
cat("Begrenset Stop Loss,","A =",opt_par$par[1],"B =",opt_par$par[2],"\n")

opt_par$par[1]+opt_par$par[2]
sort(ndata)[length(ndata)*0.99]

cat("Enkel Stop Loss, D =",opt_par_d$maximum,"\n")
##################################################
```

### Simulerer pareto porteføljer

```
m = 100000

par1 = pareto_data(m=m,alpha=2.1,beta=0.7)[[1]]
par2 = pareto_data(m=m,alpha=2.7,beta=0.7)[[1]]
par3 = pareto_data(m=m,alpha=3.5,beta=0.7)[[1]]

par4 = pareto_data(m=m,alpha=2.1,beta=1)[[1]]
par5 = pareto_data(m=m,alpha=2.1,beta=1.5)[[1]]
par6 = pareto_data(m=m,alpha=2.1,beta=2)[[1]]

par7 = pareto_data(m=m,alpha=2.1,beta=1)[[1]]
par8 = pareto_data(m=m,alpha=2.7,beta=1.5)[[1]]
par9 = pareto_data(m=m,alpha=3.5,beta=2)[[1]]
```

### Tetthetsplot for hver porteføljе

```
par(mfrow=c(1,3))

plot(density(par1),col="red",lwd=2,main="",xlab="Aggregerte krav",xlim=c(0,300),
ylim=c(0,0.085))
rect(par("usr")[1],par("usr")[3],par("usr")[2],par("usr")[4],col = "ivory")
points(density(par1),type="l",col="red",lwd=2)
lines(density(par2),col="blue",lwd=2)
lines(density(par3),col="green",lwd=2)
legend("topright",c(p1,p2,p3),fill=c("red","blue","green"),cex=1,bty = "n")
abline(h=0)

plot(density(par4),col="red",lwd=2,main="",xlab="Aggregerte krav",ylim=c(0,0.07),
xlim=c(0,350))
rect(par("usr")[1],par("usr")[3],par("usr")[2],par("usr")[4],col = "ivory")
points(density(par4),type="l",col="red",lwd=2)
lines(density(par5),col="blue",lwd=2)
lines(density(par6),col="green",lwd=2)
legend("topright",c(p4,p5,p6),fill=c("red","blue","green"),cex=1,bty = "n")
abline(h=0)

plot(density(par7),col="red",lwd=2,main="",xlab="Aggregerte krav",ylim=c(0,0.07),
xlim=c(0,350))
rect(par("usr")[1],par("usr")[3],par("usr")[2],par("usr")[4],col = "ivory")
points(density(par7),type="l",col="red",lwd=2)
lines(density(par8),col="blue",lwd=2)
lines(density(par9),col="green",lwd=2)
legend("topright",c(p7,p8,p9),fill=c("red","blue","green"),cex=1,bty = "n")
abline(h=0)

p1 = as.expression(bquote(alpha~"= 2.1,"~beta~" = 0.7"))
p2 = as.expression(bquote(alpha~"= 2.7,"~beta~" = 0.7"))
p3 = as.expression(bquote(alpha~"= 3.5,"~beta~" = 0.7"))
```

```
p4 = as.expression(bquote(alpha~"= 2.1,"~beta~" = 1"))
p5 = as.expression(bquote(alpha~"= 2.1,"~beta~" = 1.5"))
p6 = as.expression(bquote(alpha~"= 2.1,"~beta~" = 2"))
p7 = as.expression(bquote(alpha~"= 2.1,"~beta~" = 1"))
p8 = as.expression(bquote(alpha~"= 2.7,"~beta~" = 1.5"))
p9 = as.expression(bquote(alpha~"= 3.5,"~beta~" = 2"))
```

### Beregner optimal kontrakt

```
ndata_par=par9

#Startverdier
per1 = (max(ndata_par)-min(ndata_par))/3

#optimerer begrenset stop loss
opt_par = optim(c(mean(ndata_par),per1),pi_div_q_lim,X_agg=ndata_par,
control=list(fnscale=-1))

#Optimerer Enkel Stop Loss
opt_par_d = optimize(pi_div_q_enkel,interval=c(min(ndata_par),max(ndata_par)),
X_agg=ndata_par,maximum=TRUE)
```

### Utskrift av parameterene

```
cat("99 og min,",sort(ndata_par)[length(ndata_par)*0.99],"  ",min(ndata_par),"\n")
cat("Mean og SD,",mean(ndata_par),"  ",sd(ndata_par),"\n")
cat("Begrenset Stop Loss,","A =",opt_par$par[1],"B =",opt_par$par[2],"\n")

opt_par$par[1]+opt_par$par[2]
sort(ndata_par)[length(ndata_par)*0.99]

cat("Enkel Stop Loss, D =",opt_par_d$maximum,"\n")
##################################################
```

### Simulerer stokastiske porteføljer

```
m = 100000

stok1 = stokastisk_data(m=m,xi=xi,standard_alpha=2,alpha=3,beta=0.7)[[1]]
stok2 = stokastisk_data(m=m,xi=xi,standard_alpha=3,alpha=3,beta=0.7)[[1]]
stok3 = stokastisk_data(m=m,xi=xi,standard_alpha=4,alpha=3,beta=0.7)[[1]]

stok4 = stokastisk_data(m=m,xi=xi,standard_alpha=2,alpha=3,beta=0.7)[[1]]
```

```
stok5 = stokastisk_data(m=m,xi=xi,standard_alpha=3,alpha=4,beta=1)[[1]]
stok6 = stokastisk_data(m=m,xi=xi,standard_alpha=4,alpha=5,beta=1.3)[[1]]

stok7 = stokastisk_data(m=m,xi=xi,standard_alpha=2,alpha=3,beta=0.7)[[1]]
stok8 = stokastisk_data(m=m,xi=xi,standard_alpha=2,alpha=4,beta=1)[[1]]
stok9 = stokastisk_data(m=m,xi=xi,standard_alpha=2,alpha=5,beta=1.3)[[1]]
```

### Tetthetsplot for hver portefølje

```
stokp1=as.expression(psi~"= 2,"~alpha~"= 3,"~beta~"= 0.7")
stokp2=as.expression(psi~"= 3,"~alpha~"= 3,"~beta~"= 0.7")
stokp3=as.expression(psi~"= 4,"~alpha~"= 3,"~beta~"= 0.7")
stokp4=as.expression(psi~"= 2,"~alpha~"= 3,"~beta~"= 0.7")
stokp5=as.expression(psi~"= 3,"~alpha~"= 4,"~beta~"= 1")
stokp6=as.expression(psi~"= 4,"~alpha~"= 5,"~beta~"= 1.3")
stokp7=as.expression(psi~"= 2,"~alpha~"= 3,"~beta~"= 0.7")
stokp8=as.expression(psi~"= 2,"~alpha~"= 4,"~beta~"= 1")
stokp9=as.expression(psi~"= 2,"~alpha~"= 5,"~beta~"= 1.3")

par(mfrow=c(1,3))

plot(density(stok1),col="red",lwd=2,main="",xlab="Aggregerte krav",xlim=c(0,1250),
ylim=c(0,0.0045))
rect(par("usr")[1],par("usr")[3],par("usr")[2],par("usr")[4],col = "ivory")
points(density(stok1),type="l",col="red",lwd=2)
lines(density(stok2),col="blue",lwd=2)
lines(density(stok3),col="green",lwd=2)
legend("topright",c(stokp1,stokp2,stokp3),fill=c("red","blue","green"),cex=1,bty = "n")
abline(h=0)

plot(density(stok4),col="red",lwd=2,main="",xlab="Aggregerte krav",xlim=c(0,2500),
ylim=c(0,0.0045))
rect(par("usr")[1],par("usr")[3],par("usr")[2],par("usr")[4],col = "ivory")
points(density(stok4),type="l",col="red",lwd=2)
lines(density(stok5),col="blue",lwd=2)
lines(density(stok6),col="green",lwd=2)
legend("topright",c(stokp4,stokp5,stokp6),fill=c("red","blue","green"),cex=1,bty = "n")
abline(h=0)

plot(density(stok7),col="red",lwd=2,main="",xlab="Aggregerte krav",xlim=c(0,2500),
ylim=c(0,0.0045))
rect(par("usr")[1],par("usr")[3],par("usr")[2],par("usr")[4],col = "ivory")
points(density(stok7),type="l",col="red",lwd=2)
lines(density(stok8),col="blue",lwd=2)
lines(density(stok9),col="green",lwd=2)
legend("topright",c(stokp7,stokp8,stokp9),fill=c("red","blue","green"),cex=1,bty = "n")
abline(h=0)
```

### Beregning av optimal kontrakt

```
ndata_stok=stok1

#startverdier
per1 = (max(ndata_stok)-min(ndata_stok))/3

#Optimerer AxB kontrakt Method CG
opt_par = optim(c(mean(ndata_stok),per1),pi_div_q_lim,X_agg=ndata_stok,
control=list(fnscale=-1))

#Optimerer Enkel Stop Loss
opt_par_d = optimize(pi_div_q_enkel,interval=c(min(ndata_stok),max(ndata_stok)),
X_agg=ndata_stok,maximum=TRUE)
```

### Utskrift av estimatene

```
cat("99 og min,",sort(ndata_stok)[length(ndata_stok)*0.99],"  ",min(ndata_stok),"\n")
cat("Mean og SD,",mean(ndata_stok),"  ",sd(ndata_stok),"\n")
cat("Begrenset Stop Loss,","A =",opt_par$par[1],"B =",opt_par$par[2],"\n")

opt_par$par[1]+opt_par$par[2]
sort(ndata_stok)[length(ndata_stok)*0.99]

cat("Enkel Stop Loss, D =",opt_par_d$maximum,"\n")
#####################################################
```

### Sammenheng plot

```
a=c()
b=c()

sdF=c()
EF=c()

par(mfrow=c(2,2))
plot(b,main="",col="blue",type=o)
lines(sdF,col="green",type=o)

plot(b,main="",col="blue",type=o)
lines(EF,col="green",type=o)

plot(a,main="",col="blue",type=o)
lines(sdF,col="green",type=o)

plot(a,main="",col="blue",type=o)
lines(EF,col="green",type=o)

legend("topleft",col=c("blue","green"),c("a","EF"),cex=0.8,lty=1)
```

## Egendefinerte funksjoner brukt i kapittelet

```
#####################################################################
# Generer en samling av aggregerte krav med Gammafordelte krav #
#####################################################################

gamma_data = function(m,alpha,beta){

X_alle = rep(0,m)

for(i in 1:m){
#lambda=rpois(1,100)
N = rpois(1,100)
Z = rgamma(n=N,shape=alpha,scale=beta)
X_alle[i] = sum(Z)

}
return(list(X_alle))
}

#####################################################################
# Generer en samling av aggregerte krav med normalfordelte krav #
#####################################################################

normal_data = function(m,mu,theta){

X_alle = rep(0,m)

for(i in 1:m){
#lambda=rpois(1,100)
N = rpois(1,100)
Z = exp(mu + theta*rnorm(n=N,mean=0,sd=1))
X_alle[i] = sum(Z)
}
return(list(X_alle))

}

#####################################################################
# Generer en samling av aggregerte krav med paretofordelte krav #
#####################################################################

pareto_data = function(m,alpha,beta){

X_alle = rep(0,m)

for(i in 1:m){
N = rpois(1,100)
Z = rpareto(n=N,alpha=alpha,beta=beta)[[1]]
X_alle[i] = sum(Z)
```

```
}
return(list(X_alle))
}

#pareto generator

rpareto = function(n,alpha,beta){

U=runif(n)
Z = beta*(U^(-1/alpha)-1)
return(list(Z))
}


#########################################################
# Generer en samling av aggregerte krav                 #
# med gammafordelte skader med stokastisk intensitet #
#########################################################

stokastisk_data = function(m,xi,standard_alpha,alpha,beta){

X_alle = rep(0,m)
mu = xi*rgamma(m,standard_alpha)/standard_alpha
J=1000
T=1

for(i in 1:m){

N = rpois(1,J*mu[i]*T)
      Z = rgamma(n=N,shape=alpha,scale=beta)
X_alle[i] = sum(Z)

}
return(list(X_alle,mu))
}
```

```
----------
Kapittel 4
----------
```

```
ptm <- proc.time()
```

**Beregner sanne verdier for gamma fordelte krav**

```
sann_verdi_nor=sannhet(J=J,int=int,type="normal")
Fe_enkel_sann_nor=sann_verdi_nor[[1]];Fe_begrenset_sann_nor=sann_verdi_nor[[2]]
sann_data_nor=sann_verdi_nor[[3]];a_sann_nor=sann_verdi_nor[[4]]
```

```
b_sann_nor=sann_verdi_nor[[5]];ae_sann_nor=sann_verdi_nor[[6]]
```

### Beregner sanne verdier for gamma fordelte krav

```
sann_verdi_gam=sannhet(J=J,int=int,type="gamma")
Fe_enkel_sann_gam=sann_verdi_gam[[1]];Fe_begrenset_sann_gam=sann_verdi_gam[[2]]
sann_data_gam=sann_verdi_gam[[3]];a_sann_gam=sann_verdi_gam[[4]]
b_sann_gam=sann_verdi_gam[[5]];ae_sann_gam=sann_verdi_gam[[6]
```

```
ptm <- proc.time()
```

### Estimerer fra historisk data og genererer bootstrap estimater av a og b

```
param=genAogB(A=A,mb=mb,m=m,int=int,J=J)
a_boot_nor=param[[1]];b_boot_nor=param[[2]];a_boot_enkel_nor=param[[3]]
a_boot_gam=param[[4]];b_boot_gam=param[[5]];a_boot_enkel_gam=param[[6]]
```

### Printer ut mellomresultater

```
print_ab(a=a_sann_nor,b=b_sann_nor,c=ae_sann_nor,d=mean(a_boot_nor),
e=mean(a_boot_nor),f=mean(a_boot_enkel_nor),g=a_sann_gam,h=b_sann_gam,
i=ae_sann_gam,j=mean(a_boot_gam),k=mean(b_boot_gam),l=mean(a_boot_enkel_gam))
```

### Estimerer fra historisk data og genererer bootstrap estimater av a og b

```
kriterie_bt=bootstrap_kriterie(a_boot_nor,b_boot_nor,a_boot_enkel_nor,
a_boot_gam,b_boot_gam,a_boot_enkel_gam)
```

```
proc.time() - ptm
```

### Egendefinerte funksjoner brukt i kapittelet

---

```
##########################################################################
# Beregning av estimat feil og returnering av alle relevante estimater #
##########################################################################

feil_plot_pi_enkel_nor=function(Fe_enkel_sann_nor,Fe_begrenset_sann_nor,
Fe_enkel_sann_gam,Fe_begrenset_sann_gam){

A=1000000
m_list=c(25,50,100,1000,10000,100000,1000000)
mb=100

  pi_e_nor=rep(0,length(m_list))
  pi_b_nor=rep(0,length(m_list))
  pi_e_gam=rep(0,length(m_list))
```

```
  pi_b_gam=rep(0,length(m_list))

  sd_e_nor=rep(0,length(m_list))
  sd_b_nor=rep(0,length(m_list))
  sd_e_gam=rep(0,length(m_list))
  sd_b_gam=rep(0,length(m_list))

for(j in 1:length(m_list)){

#Generer bootstrap estimater av a og b
param=genAogB(A=A,mb=mb,m=m_list,int=0.5,J=1000)
a_boot_nor=param[[1]];b_boot_nor=param[[2]];a_boot_enkel_nor=param[[3]]
a_boot_gam=param[[4]];b_boot_gam=param[[5]];a_boot_enkel_gam=param[[6]]

#printout
print_ab(a=a_sann_nor,b=b_sann_nor,c=ae_sann_nor,d=mean(a_boot_nor),e=mean(a_boot_nor),
f=mean(a_boot_enkel_nor),g=a_sann_gam,h=b_sann_gam,i=ae_sann_gam,j=mean(a_boot_gam),
k=mean(b_boot_gam),l=mean(a_boot_enkel_gam))

#Bootstrap kriterie
kriterie_bt=bootstrap_kriterie(a_boot_nor,b_boot_nor,a_boot_enkel_nor,
a_boot_gam,b_boot_gam,a_boot_enkel_gam)

Fe_begrenset_bt_nor=kriterie_bt[[1]];Fe_enkel_bt_nor=kriterie_bt[[2]]
Fe_begrenset_bt_gam=kriterie_bt[[3]];Fe_enkel_bt_gam=kriterie_bt[[4]]

#Beregner feil
pi_e_nor=Fe_enkel_sann_nor-mean(Fe_enkel_bt_nor)
pi_b_nor=Fe_begrenset_sann_nor-mean(Fe_begrenset_bt_nor)
pi_e_gam=Fe_enkel_sann_gam-mean(Fe_enkel_bt_gam)
pi_b_gam=Fe_begrenset_sann_gam-mean(Fe_begrenset_bt_gam)

sd_e_nor=sd(Fe_enkel_bt_nor)
sd_b_nor=sd(Fe_begrenset_bt_nor)
sd_e_gam=sd(Fe_enkel_bt_gam)
sd_b_gam=sd(Fe_begrenset_bt_gam)

}
alarm()
return(list(pi_e_nor,pi_b_nor,pi_e_gam,pi_b_gam,m_list,a_boot_nor,
b_boot_nor,a_boot_gam,b_boot_gam,sd_e_nor,sd_b_nor,sd_e_gam,sd_b_gam))
}

#################################################################
# Plot av aktuelle verdier som returneres fra funksjonen over #
#################################################################

par(mfrow=c(1,2))

jamanmb=c(25,500,1000,5000,10000,50000)
```

```
plot(log(jamanmb),lwd=3,xlab="antall simuleringer m",jamanplot[[6]][1:6],
ylab="a og b verdi",xaxt = "n",col="blue",type="o",
main=as.expression(bquote("Normal")))

rect(par("usr")[1],par("usr")[3],par("usr")[2],par("usr")[4],col = "ivory")
lines(log(jamanmb),lwd=3,jamanplot[[10]][1:6],ylim=c(0,0.025),xaxt = "n",
col="blue",type="o")
axis(1,at=c(log(jamanmb)[1:6]),labels=c("25","500","1000","5000","10000","50000"))

lines(log(jamanmb),lwd=3,xlab="antall bootstrapreplikasjoner",jamanplot[[7]][1:6],
xaxt = "n",col="green",type="o",main=as.expression(bquote("Normal")))
axis(1,at=c(log(jamanmb)[1:6]),labels=c("25","500","1000","5000","10000","50000"))
legend("topright",col=c("blue","green"),c("enkel stop-loss","begrenset stop-loss"),
cex=0.7,lty=1,lwd=2,bty="n")

plot(log(jamanmb),lwd=3,xlab="antall simuleringer m",jamanplot[[8]][1:6],
ylab="a og b verdi",xaxt = "n",col="blue",type="o",main=as.expression(bquote("Gamma")))
axis(1,at=c(log(jamanmb)[1:6]),labels=c("25","500","1000","5000","10000","50000"))
rect(par("usr")[1],par("usr")[3],par("usr")[2],par("usr")[4],col = "ivory")
lines(log(jamanmb),lwd=3,jamanplot[[12]][1:6],xaxt = "n",col="blue",type="o")

lines(log(jamanmb),lwd=3,xlab="antall bootstrap replikasjoner",jamanplot[[9]][1:6],
xaxt = "n",col="green",type="o",main=as.expression(bquote("begrenset gamma")))
axis(1,at=c(log(jamanmb)[1:6]),labels=c("25","500","1000","5000","10000","50000"))
legend("topright",col=c("blue","green"),c("enkel stop-loss","begrenset stop-loss"),
cex=0.7,lty=1,lwd=2,bty="n")


########################################
# Printe funksjon for beregnede verdier #
########################################

print_ab=function(a=a_sann_nor,b=b_sann_nor,c=ae_sann_nor,
d=mean(a_boot_nor),e=mean(a_boot_nor),f=mean(a_boot_enkel_nor),g=a_sann_gam,
h=b_sann_gam,i=ae_sann_gam,j=mean(a_boot_gam),k=mean(b_boot_gam),
l=mean(a_boot_enkel_gam)){

cat("-----------------------------------------------------","\n")
cat("Sann Normal:a =",a,"b =",b,"a_e =",c,"\n")
cat("Normal:     a =",d,"b =",e,"a_e =",f,"\n")
cat("Sann Gamma: a =",g,"b =",h,"a_e =",i,"\n")
cat("Gamma:      a =",j,"b =",k,"a_e =",l,"\n")
cat("-----------------------------------------------------","\n")


}

#########################################################################
# Funksjon som beregner og returnerer bootstrap estimater av kriterie #
#########################################################################
```

```
bootstrap_kriterie=function(a_boot_nor,b_boot_nor,a_boot_enkel_nor,
a_boot_gam,b_boot_gam,a_boot_enkel_gam){

Fe_enkel_bt_nor=rep(0,length(a_boot_nor))
Fe_begrenset_bt_nor=rep(0,length(a_boot_nor))
Fe_enkel_bt_gam=rep(0,length(a_boot_gam))
Fe_begrenset_bt_gam=rep(0,length(a_boot_gam))

for(i in 1:length(a_boot_nor)){

Fe_begrenset_bt_nor[i]=pi_div_q_lim(c(a_boot_nor[i],b_boot_nor[i]),sann_data_nor)
Fe_enkel_bt_nor[i]=pi_div_q_enkel(a_boot_enkel_nor[i],sann_data_nor)

}

for(i in 1:length(a_boot_gam)){

Fe_begrenset_bt_gam[i]=pi_div_q_lim(c(a_boot_gam[i],b_boot_gam[i]),X_agg=sann_data_gam)
Fe_enkel_bt_gam[i]=pi_div_q_enkel(a_boot_enkel_gam[i],X_agg=sann_data_gam)

}
cat("--------------------------------------","\n")
cat("Normal feil","\n")
cat("--------------------------------------","\n")
cat("Enkel:",Fe_enkel_sann_nor-mean(Fe_enkel_bt_nor),"","\n")
cat("Begrenset:",Fe_begrenset_sann_nor-mean(Fe_begrenset_bt_nor),"\n")
cat("Sd Enkel:",sd(Fe_enkel_bt_nor),"","\n")
cat("Sd Begrenset:",sd(Fe_begrenset_bt_nor),"\n")
cat("--------------------------------------","\n")
cat("Gamma feil","\n")
cat("--------------------------------------","\n")
cat("Enkel:",Fe_enkel_sann_gam-mean(Fe_enkel_bt_gam),"\n")
cat("Begrenset:",Fe_begrenset_sann_gam-mean(Fe_begrenset_bt_gam),"\n")
cat("Sd Enkel:",sd(Fe_enkel_bt_gam),"","\n")
cat("Sd Begrenset:",sd(Fe_begrenset_bt_gam),"\n")


alarm()

return(list(Fe_begrenset_bt_nor,Fe_enkel_bt_nor,Fe_begrenset_bt_gam,Fe_enkel_bt_gam))

}

##############################################################################
# Funksjon som beregner og returerner sanne verdier for a, b og tilhorende kriterie #
##############################################################################

sannhet = function(J,int,type){

m=1000000
```

```
if(type=="normal"){
print("normal bitchez")
X_alle=rep(0,m)
for(i in 1:m){
N = rpois(1,J*int)
Z = exp(0.5 + 0.8*rnorm(n=N,mean=0,sd=1))
X_alle[i] = sum(Z)
}

} else {

print("gamma bitchez")
X_alle=rep(0,m)
for(i in 1:m){
N = rpois(1,J*int)
Z = rgamma(n=N,shape=1,scale=2)
X_alle[i] = sum(Z)
}


}

ndata=X_alle
per1 = (max(ndata)-min(ndata))/3

#Optimerer Begrenset Stop Loss
opt_par = optim(c(mean(ndata),per1),pi_div_q_lim,X_agg=ndata,
control=list(fnscale=-1))

#Optimerer Enkel Stop Loss
opt_par_d = optimize(pi_div_q_enkel,interval=c(min(ndata),max(ndata)),
X_agg=ndata,maximum=TRUE)

Fe_enkel=pi_div_q_enkel(opt_par_d$maximum,X_agg=ndata)
Fe_begrenset=pi_div_q_lim(kon=c(opt_par$par[1],opt_par$par[2]),X_agg=ndata)

if(round(sort(ndata)[length(ndata)*0.99])==round(opt_par$par[1]+opt_par$par[2])){
cat("----------","\n")
cat("Good to go","\n")
cat("----------","\n")
cat(sort(ndata)[length(ndata)*0.99],"\n")
cat(round(opt_par$par[1]+opt_par$par[2]),"\n")
cat("----------","\n")
cat("A =",opt_par$par[1],"\n")
cat("B =",opt_par$par[2],"\n")



} else {
cat(sort(ndata)[length(ndata)*0.99],round(opt_par$par[1]+opt_par$par[2]),"\n")
cat("----------","\n")
```

```
cat(sort(ndata)[length(ndata)*0.99],"\n")
cat(round(opt_par$par[1]+opt_par$par[2]),"\n")
cat("----------","\n")
cat("A =",opt_par$par[1],"\n")
cat("B =",opt_par$par[2],"\n")


      }

a=opt_par$par[1]
b=opt_par$par[2]
a_e=opt_par_d$max

alarm()

return(list(Fe_enkel,Fe_begrenset,X_alle,a,b,a_e))

}



######################################################################
# Funksjon som estimerer fra historisk data, beregner bootstrap #
# estimatene og returnere bootstrap estimater av kriterie       #
######################################################################


genAogB = function(A,mb,m,int,J){

#Historisk data
A=A
J=J

N = rpois(n=1,lambda=A*int)
Z_nor = exp(0.5 + 0.8*rnorm(n=N,mean=0,sd=1))
Z_gam = rgamma(n=N,shape=1,scale=2)


#Estimater

mu_hat=N/A
alpha_hat_nor=mean(log(Z_nor))
beta_hat_nor=mean=sd(log(Z_nor))
alpha_hat_gam=mean((mean(Z_gam)/sd(Z_gam))^2)
beta_hat_gam=mean(Z_gam)

cat("----------------------------------------","\n")
cat("n =",N,"A =",A,"m =",m,"J =",J,"mb =",mb,"\n")
cat("----------------------------------------","\n")
cat("Historiske estimater","\n")
cat("-----------------------------------","\n")
```

```
cat("alpha normal ->",alpha_hat_nor,"\n")
cat("beta normal ->",beta_hat_nor,"\n")
cat("alpha gamma ->",alpha_hat_gam,"\n")
cat("beta gamma ->",beta_hat_gam,"\n")
cat("mu ->",mu_hat,"\n")

#bootstrap replikasjoner
mb=mb

#Poisson bootstrap
mu_st=rpois(mb,A*mu_hat)/A

#Normal bootstrap
Z_st_nor=matrix(rlnorm(N*mb,alpha_hat_nor,beta_hat_nor),N,mb)

#Gamma boostrap
Z_st_gam=matrix(rgamma(N*mb,shape=alpha_hat_gam,scale=beta_hat_gam),N,mb)

#Bootstrap estimater
alpha_st_nor=apply(log(Z_st_nor),2,mean)
beta_st_nor=apply(log(Z_st_nor),2,sd)

alpha_st_gam=rep(0,mb)
beta_st_gam=rep(0,mb)
for(i in 1:mb){
alpha_st_gam[i]=mean((mean(Z_st_gam[,i])/sd(Z_st_gam[,i]))^2)
beta_st_gam[i]=mean(Z_st_gam[,i])
}


cat("-------------------------------------","\n")
cat("Bootstrap estimater - forventet verdi","\n")
cat("-------------------------------------","\n")
cat("alpha normal ->",mean(alpha_st_nor),"\n")
cat("beta normal ->",mean(beta_st_nor),"\n")
cat("alpha gamma ->",mean(alpha_st_gam),"\n")
cat("beta gamma ->",mean(beta_st_gam),"\n")
cat("mu ->",mean(mu_st),"\n")
cat("-------------------------------------","\n")

#Monte Carlo
m=m
X_agg_nor=matrix(0,mb,m)
X_agg_gam=matrix(0,mb,m)
for(i in 1:mb){
for(j in 1:m){
N = rpois(1,1000*mu_st[i])
Z_nor = exp(alpha_st_nor[i] + beta_st_nor[i]*rnorm(n=N,mean=0,sd=1))
Z_gam = rgamma(N,shape=alpha_st_gam[i],scale=beta_st_gam[i])
X_agg_nor[i,j] = sum(Z_nor)
X_agg_gam[i,j] = sum(Z_gam)
```

```
}

}

#Regner a og b bare normal
a_st_nor=rep(0,mb)
b_st_nor=rep(0,mb)
a_st_enkel_nor=rep(0,mb)
a_st_gam=rep(0,mb)
b_st_gam=rep(0,mb)
a_st_enkel_gam=rep(0,mb)

for(i in 1:mb){
ndata=X_agg_nor[i,]
   per1 = (max(ndata)-min(ndata))/3
#Optimerer Begrenset Stop Loss
opt_par = optim(c(mean(ndata),per1),pi_div_q_lim,X_agg=ndata,control=list(fnscale=-1))
#Optimerer Enkel Stop Loss
opt_par_d = optimize(pi_div_q_enkel,interval=c(min(ndata),max(ndata)),
X_agg=ndata,maximum=TRUE)

if(round(sort(ndata)[length(ndata)*0.99])==round(opt_par$par[1]+opt_par$par[2])){
a_st_nor[i]=opt_par$par[1]
b_st_nor[i]=opt_par$par[2]
a_st_enkel_nor[i]=opt_par_d$max

      } else {

a_st_nor[i]=0
b_st_nor[i]=0
a_st_enkel_nor[i]=0

      }

ndata=X_agg_gam[i,]
   per1 = (max(ndata)-min(ndata))/3
#Optimerer Begrenset Stop Loss
opt_par = optim(c(mean(ndata),per1),pi_div_q_lim,X_agg=ndata,control=list(fnscale=-1))
#Optimerer Enkel Stop Loss
opt_par_d = optimize(pi_div_q_enkel,interval=c(min(ndata),max(ndata)),
X_agg=ndata,maximum=TRUE)

if(round(sort(ndata)[length(ndata)*0.99])==round(opt_par$par[1]+opt_par$par[2])){
a_st_gam[i]=opt_par$par[1]
b_st_gam[i]=opt_par$par[2]
a_st_enkel_gam[i]=opt_par_d$max

      } else {

a_st_gam[i]=0
```

```
b_st_gam[i]=0
a_st_enkel_gam[i]=0

      }


}

a_st_nor=a_st_nor[a_st_nor!=0]
b_st_nor=b_st_nor[b_st_nor!=0]
a_st_enkel_nor=a_st_enkel_nor[a_st_enkel_nor!=0]

a_st_gam=a_st_gam[a_st_gam!=0]
b_st_gam=b_st_gam[b_st_gam!=0]
a_st_enkel_gam=a_st_enkel_gam[a_st_enkel_gam!=0]


alarm()

return(list(a_st_nor,b_st_nor,a_st_enkel_nor,a_st_gam,b_st_gam,
a_st_enkel_gam,X_agg_nor,X_agg_gam))

}
```

**Leser inn datafiler simulert fra Fortran, skriver ut resultater og plotter**

---

```
sanngamA=304.327;sanngamB=405.840-sanngamA;sanngamF=0.0888867
sannnorA=101.008;sannnorB=127.625-sannnorA;sannnorF=0.0910477

#Leser inn data variert A
##########################
MvarA=readinvarA()
MvarAnor=MvarA[[1]];MvarAgam=MvarA[[2]]

#Leser inn data variert M
##########################
MvarM=readinvarM()
MvarMnor=MvarM[[1]];MvarMgam=MvarM[[2]]

#Leser inn data variert MB
###########################
MvarMB=readinvarMB()
MvarMBgam5=MvarMB[[1]];MvarMBgam10=MvarMB[[2]];MvarMBgam25=MvarMB[[3]]
MvarMBgam50=MvarMB[[4]];MvarMBgam100=MvarMB[[5]];MvarMBgam1000=MvarMB[[6]]

MvarMBnor5=MvarMB[[7]];MvarMBnor10=MvarMB[[8]];MvarMBnor25=MvarMB[[9]]
MvarMBnor50=MvarMB[[10]];MvarMBnor100=MvarMB[[11]];MvarMBnor1000=MvarMB[[12]]

#Lager tabeller
```

```
###############

#A
for(i in 1:7){cat("&",sanngamF-mean(MvarAgam[i,]),"&",sannnorF-mean(MvarAnor[i,]),
"\\","\n")}

#M

for(i in 1:6){cat("&",sanngamF-mean(MvarMgam[i,]),"&",sannnorF-mean(MvarMnor[i,]),
"\\","\n")}

#MB

for(i in 1:1){
cat("&",sanngamF-mean(MvarMBgam5),"&",sannnorF-mean(MvarMBnor5),"\\","\n")
cat("&",sanngamF-mean(MvarMBgam10),"&",sannnorF-mean(MvarMBnor10),"\\","\n")
cat("&",sanngamF-mean(MvarMBgam25),"&",sannnorF-mean(MvarMBnor25),"\\","\n")
cat("&",sanngamF-mean(MvarMBgam50),"&",sannnorF-mean(MvarMBnor50),"\\","\n")
cat("&",sanngamF-mean(MvarMBgam100),"&",sannnorF-mean(MvarMBnor100),"\\","\n")
cat("&",sanngamF-mean(MvarMBgam1000),"&",sannnorF-mean(MvarMBnor1000),"\\","\n")
}

#Plotte
#######

#M - SD(F)
##########

m=c(100,1000,10000,100000,500000,1000000)
Fsdgam=rep(0,length(m))
Fsdnor=rep(0,length(m))

for(i in 1:length(m)){Fsdgam[i]=sd(MvarMgam[i,]);Fsdnor[i]=sd(MvarMnor[i,])}

plot(log(m),Fsdgam,lwd=3,xlab="antall simuleringer",ylab="sd(F)",ylim=c(0.002,0.012),
,xaxt = "n",col="blue",type="o",main=as.expression(bquote("")))
rect(par("usr")[1],par("usr")[3],par("usr")[2],par("usr")[4],col = "ivory")
lines(log(m),Fsdgam,lwd=3,xaxt = "n",col="blue",type="o")
lines(log(m),Fsdnor,lwd=3,xaxt = "n",col="green",type="o")
axis(1,at=c(log(m)),labels=c("100","1000","10000","100000","500000","1000000"))
legend("topright",col=c("blue","green"),c("Gamma","Normal"),cex=1,lty=1,lwd=2,bty="n")


#A - SD(F)
##########

Fsdgam=rep(0,7)
Fsdnor=rep(0,7)
for(i in 1:7){Fsdgam[i]=sd(MvarAgam[i,]);Fsdnor[i]=sd(MvarAnor[i,])}

A=c(100,500,1000,5000,10000,50000,100000)
```

```
plot(log(A),Fsdnor,lwd=3,xlab="antall historiske skader",ylab="sd(F)",
ylim=c(min(Fsdnor),max(Fsdgam)),
,xaxt = "n",col="blue",type="o",main=as.expression(bquote("")))
rect(par("usr")[1],par("usr")[3],par("usr")[2],par("usr")[4],col = "ivory")
lines(log(A),Fsdgam,lwd=3,xaxt = "n",col="blue",type="o")
lines(log(A),Fsdnor,lwd=3,xaxt = "n",col="green",type="o")
axis(1,at=c(log(A)),labels=c("100","500","1000","5000","10000","50000","100000"))
legend("topright",col=c("blue","green"),c("Gamma","Normal"),cex=1,lty=1,lwd=2,bty="n")



#####################################
# Leser in datafiler der MB varieres #
#####################################

readinvarMB = function(){

      varMB = c(5,10,25,50,100,1000)

MvarMBnor5 = rep(0,5)
MvarMBnor10 = rep(0,10)
MvarMBnor25 = rep(0,25)
MvarMBnor50 = rep(0,50)
MvarMBnor100 = rep(0,100)
MvarMBnor1000 = rep(0,1000)


#Normal MvarMB
for (i in 1:length(varMB)) {

if(i==1){
temp=scan(paste("varMB",format(varMB[i],scientific=FALSE),"nor.txt",sep=""))
print(paste("varMB",format(varMB[i],scientific=FALSE),"nor.txt",sep=""))
temp=temp[4:length(temp)]
temp1=rep(0,(length(temp)/3))
for(j in 1:length(temp1)){temp1[j]=temp[j*3]}
MvarMBnor5 = temp1[1:length(temp1)]
}
else if(i==2) {
temp=scan(paste("varMB",format(varMB[i],scientific=FALSE),"nor.txt",sep=""))
print(paste("varMB",format(varMB[i],scientific=FALSE),"nor.txt",sep=""))
temp=temp[4:length(temp)]
temp1=rep(0,(length(temp)/3))
for(j in 1:length(temp1)){temp1[j]=temp[j*3]}
MvarMBnor10 = temp1[1:length(temp1)]

}
else if(i==3) {
temp=scan(paste("varMB",format(varMB[i],scientific=FALSE),"nor.txt",sep=""))
print(paste("varMB",format(varMB[i],scientific=FALSE),"nor.txt",sep=""))
```

```
temp=temp[4:length(temp)]
temp1=rep(0,(length(temp)/3))
for(j in 1:length(temp1)){temp1[j]=temp[j*3]}
MvarMBnor25 = temp1[1:length(temp1)]

}
else if(i==4) {
temp=scan(paste("varMB",format(varMB[i],scientific=FALSE),"nor.txt",sep=""))
print(paste("varMB",format(varMB[i],scientific=FALSE),"nor.txt",sep=""))

temp=temp[4:length(temp)]
temp1=rep(0,(length(temp)/3))
for(j in 1:length(temp1)){temp1[j]=temp[j*3]}
MvarMBnor50 = temp1[1:length(temp1)]

}
else if(i==5) {
temp=scan(paste("varMB",format(varMB[i],scientific=FALSE),"nor.txt",sep=""))
print(paste("varMB",format(varMB[i],scientific=FALSE),"nor.txt",sep=""))
temp=temp[4:length(temp)]
temp1=rep(0,(length(temp)/3))
for(j in 1:length(temp1)){temp1[j]=temp[j*3]}
MvarMBnor100 = temp1[1:length(temp1)]

}
else {

temp=scan(paste("varMB",format(varMB[i],scientific=FALSE),"nor.txt",sep=""))
print(paste("varMB",format(varMB[i],scientific=FALSE),"nor.txt",sep=""))
temp=temp[4:length(temp)]
temp1=rep(0,(length(temp)/3))
for(j in 1:length(temp1)){temp1[j]=temp[j*3]}
MvarMBnor1000 = temp1[1:length(temp1)]

}

}

  MvarMBgam5 = rep(0,5)
MvarMBgam10 = rep(0,10)
MvarMBgam25 = rep(0,25)
MvarMBgam50 = rep(0,50)
MvarMBgam100 = rep(0,100)
MvarMBgam1000 = rep(0,1000)


#Gamma MvarMB
for (i in 1:length(varMB)) {


if(i==1){
```

```
temp=scan(paste("varMB",format(varMB[i],scientific=FALSE),"gam.txt",sep=""))
print(paste("varMB",format(varMB[i],scientific=FALSE),"nor.txt",sep=""))

temp=temp[4:length(temp)]
temp1=rep(0,(length(temp)/3))
for(j in 1:length(temp1)){temp1[j]=temp[j*3]}
MvarMBgam5 = temp1[1:length(temp1)]
}
else if(i==2) {
temp=scan(paste("varMB",format(varMB[i],scientific=FALSE),"gam.txt",sep=""))
print(paste("varMB",format(varMB[i],scientific=FALSE),"nor.txt",sep=""))

temp=temp[4:length(temp)]
temp1=rep(0,(length(temp)/3))
for(j in 1:length(temp1)){temp1[j]=temp[j*3]}
MvarMBgam10 = temp1[1:length(temp1)]

}

else if(i==3) {
temp=scan(paste("varMB",format(varMB[i],scientific=FALSE),"gam.txt",sep=""))
print(paste("varMB",format(varMB[i],scientific=FALSE),"nor.txt",sep=""))

temp=temp[4:length(temp)]
temp1=rep(0,(length(temp)/3))
for(j in 1:length(temp1)){temp1[j]=temp[j*3]}
MvarMBgam25 = temp1[1:length(temp1)]

}
else if(i==4) {
temp=scan(paste("varMB",format(varMB[i],scientific=FALSE),"gam.txt",sep=""))
print(paste("varMB",format(varMB[i],scientific=FALSE),"nor.txt",sep=""))

temp=temp[4:length(temp)]
temp1=rep(0,(length(temp)/3))
for(j in 1:length(temp1)){temp1[j]=temp[j*3]}
MvarMBgam50 = temp1[1:length(temp1)]

}
else if(i==5) {
temp=scan(paste("varMB",format(varMB[i],scientific=FALSE),"gam.txt",sep=""))
print(paste("varMB",format(varMB[i],scientific=FALSE),"nor.txt",sep=""))

temp=temp[4:length(temp)]
temp1=rep(0,(length(temp)/3))
for(j in 1:length(temp1)){temp1[j]=temp[j*3]}
MvarMBgam100 = temp1[1:length(temp1)]

}
else {
```

```
temp=scan(paste("varMB",format(varMB[i],scientific=FALSE),"gam.txt",sep=""))
print(paste("varMB",format(varMB[i],scientific=FALSE),"nor.txt",sep=""))

temp=temp[4:length(temp)]
temp1=rep(0,(length(temp)/3))
for(j in 1:length(temp1)){temp1[j]=temp[j*3]}
MvarMBgam1000 = temp1[1:length(temp1)]

}

}

return(list(MvarMBgam5,MvarMBgam25,MvarMBgam10,MvarMBgam50,MvarMBgam100,MvarMBgam1000,
            MvarMBnor5,MvarMBgam25,MvarMBnor10,MvarMBnor50,MvarMBnor100,MvarMBnor1000))

}


####################################
# Leser in datafiler der M varieres #
####################################


readinvarM = function(){

mb=50

      varM = c(100,1000,10000,100000,500000,1000000)

MvarMnor = matrix(0,length(varM),mb)

#Normal MvarM
for (i in 1:length(varM)) {

temp=scan(paste("varM",format(varM[i],scientific=FALSE),"nor.txt",sep=""))
temp=temp[4:length(temp)]
temp1=rep(0,(length(temp)/3))
for(j in 1:length(temp1)){temp1[j]=temp[j*3]}
length(temp1)
MvarMnor[i,] = temp1[1:length(temp1)]

}

MvarMgam = matrix(0,length(varM),100)

#Gamma MvarM
for (i in 1:length(varM)) {

temp=scan(paste("varM",format(varM[i],scientific=FALSE),"gam.txt",sep=""))
temp=temp[4:length(temp)]
temp1=rep(0,(length(temp)/3))
```

```
for(j in 1:length(temp1)){temp1[j]=temp[j*3]}
length(temp1)
MvarMgam[i,] = temp1[1:length(temp1)]

}
return(list(MvarMnor,MvarMgam))

}

#####################################
# Leser in datafiler der MB varieres #
#####################################

readinvarA = function(){

varA = c(100,500,1000,5000,10000,50000,100000)

MvarAnor = matrix(0,length(varA),100)

#Normal MvarA
for (i in 1:length(varA)) {

temp=scan(paste("varA",format(varA[i],scientific=FALSE),"nor.txt",sep=""))
temp=temp[4:length(temp)]
temp1=rep(0,(length(temp)/3))
for(j in 1:length(temp1)){temp1[j]=temp[j*3]}
length(temp1)
MvarAnor[i,] = temp1[1:length(temp1)]

}

MvarAgam = matrix(0,7,100)

#Gamma MvarA
for (i in 1:length(varA)) {

temp=scan(paste("varA",format(varA[i],scientific=FALSE),"gam.txt",sep=""))
temp=temp[4:length(temp)]
temp1=rep(0,(length(temp)/3))
for(j in 1:length(temp1)){temp1[j]=temp[j*3]}
length(temp1)
MvarAgam[i,] = temp1[1:length(temp1)]

}
return(list(MvarAnor,MvarAgam))

}
```

# 7 Bibliografi

[1] Chi Y og Tan K.S. (2011). Optimal reinsurance under Var and Cvar risk measures, 1-3.

[2] K.C. Cheung, K.C.J. Sung, S.C.P. Yam og S.P. Yung. (2011). Optimal reinsurance under general law-invariant risk measures, 1-20.

[3] Bølviken E. (2014). *Computation and Modelling in Insurance and Finance*, Cambridge University Press, Cambridge.

[4] Devore J.L og Berk K.N (2007). *Modern Mathematical Statistics with applications.* Thompson Brooks/Cole, Duxbury.

[4] Efron, B. og Tibshirani, R.J.(1993). *An introduction to The Boostrap* New York: Chapman And Hall.